A Unified Framework for Coordinated Multi-Arm Motion Planning

Journal Title
XX(X):1–25

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Seyed Sina Mirrazavi Salehian[*1], Nadia Figueroa[*1] and Aude Billard[1]

Abstract
Coordination is essential in the design of dynamic control strategies for multi-arm robotic systems. Given the complexity of the task and dexterity of the system, coordination constraints can emerge from different levels of planning and control. Primarily, one must consider *task-space coordination*, where the robots must coordinate with each other, with an object or with a target of interest. Coordination is also necessary in *joint-space*, as the robots should avoid self-collisions at any time. We provide such *joint-space* coordination by introducing a centralized inverse kinematics (IK) solver under self-collision avoidance constraints; formulated as a quadratic program (QP) and solved in real-time. The space of free motion is modeled through a sparse non-linear kernel classification method in a data-driven learning approach. Moreover, we provide multi-arm *task-space* coordination for both *synchronous* or *asynchronous* behaviors. We define a *synchronous* behavior as that in which the robot arms must coordinate with each other and with a moving object such that they reach for it in *synchrony*. In contrast, an *asynchronous* behavior allows for each robot to perform independent point-to-point reaching motions. To transition smoothly from asynchronous to synchronous behaviors and conversely, we introduce the notion of *synchronization allocation*. We show how this allocation can be controlled through an external variable, such as the location of the object to be manipulated. Both behaviors and their synchronization allocation are encoded in a single dynamical system. We validate our framework on a dual-arm robotic system and demonstrate that the robots can re-synchronize and adapt the motion of each arm while avoiding self-collision within milliseconds. The speed of control is exploited to intercept fast moving objects whose motion cannot be predicted accurately.

Keywords
Multi-arm motion planning, coordination, dynamical systems, self-collision avoidance

1 Introduction

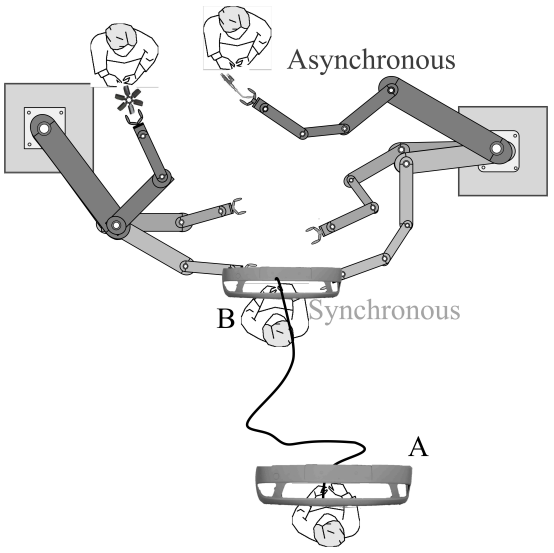
The use of multi-arm robotic systems allows for highly complex manipulation of heavy or bulky objects that would otherwise be infeasible for a single-arm robot. One can envision a plethora of applications in smart-factories or homes, that would benefit from such extended workspace and capabilities. Examples include, lifting, grabbing, catching, manipulating objects with multiple arms which could be either traveling on a cart or a running conveyor belt, carried by humans or even flying towards the multi-arm robot system, as depicted in Fig. 1a. Moreover, a multi-arm system could provide not only *synchronous* behaviors, as the ones mentioned above, but also *asynchronous* behaviors, where each robot follows its own goal-oriented task (Fig. 1a). Multi-arm control strategies endowed with these capabilities can pave the way for the flexible manufacturing systems of the future.

The challenge is then to control these robots in a *coordinated manner*, in order to *safely* and *efficiently* achieve the desired manipulation task. Due to the technological difficulties that entail *coordinating* multiple arms with a dynamic object in a computationally efficient way, these applications have yet to be explored in the robotics community. Most effort in the field of multi-arm control has focused primarily on devising strategies for coordinated manipulation of *static* objects that are partially or fully grasped by the multi-arm system (Caccavale and Uchiyama 2016). Seldom work has focused on developing *coordinated strategies* that a multi-arm system can use to reach and grab *moving* objects *synchronously*; while also being capable of reaching for independent targets or objects *asynchronously* (Vahrenkamp et al. 2012, 2010).

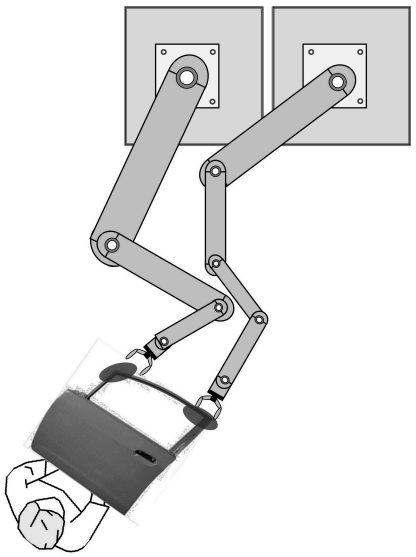
In this work, we draw inspiration from the field of human coordination dynamics, in order to devise *safe* and *coordinated* motions in dual/multi-arm robotic systems. Humans have a remarkable way of controlling their bi-manual movements in everyday life. They are capable of coordinating both arms and hands in synchronous and asynchronous tasks, in uni-manual and bi-manual tasks, and they transition smoothly between all of these behaviors. Interestingly, the motion of the hands, and accordingly of the arms, is generated in a smooth and efficient manner, all the while avoiding self-collisions between their limbs. Seminal works from the field of human coordination dynamics suggest that human inter-limb coordination is governed

[*] These authors contributed equally to this work.
[1] Learning Algorithms and Systems Laboratory (LASA), Swiss Federal Institute of Technology, Lausanne (EPFL) CH-1015 Lausanne, Switzerland.

Corresponding author:
Seyed Sina Mirrazavi Salehian, EPFL-STI-IMT-LASA, ME A3 424 (Bâtiment ME) Station 9 CH-1015 Lausanne, Switzerland
Email: sina.mirrazavi@epfl.ch



(a) Transitioning between *Synchronous* and *Asynchronous* Multi-Arm Task-Space Behavior



(b) *Self-Collision* Avoidance in Joint-Space

Figure 1. Illustration of Multi-arm Task-Space Coordination and Joint-Space Collision Avoidance. (a) *Synchronous* and *Asynchronous* task-space behaviors, where the robots coordinate with each other to simultaneously reach-for a moving object or each robot has its own target and is endowed with an independent stable DS to generate desired motions, respectively. (b) Self-Collision Avoidance (SCA) in Joint-Space for both task-space behaviors.

by a strongly coupled underlying nonlinear dynamical system (DS) (Kelso et al. 1979; Kelso 1984). These studies involved transitioning between in-phase and anti-phase rhythmic movements, which led to the Haken-Kelso-Bunz (HKB) model of coordination (Haken et al. 1985). (Swinnen 2002) highlights strong limb-coupling in terms of the *spatial* constraints that govern coordination, which can be *egocentric* (based on mirror symmetry in muscle groups) or *allocentric* (following the same direction in extrinsic space). Later on, (Calvin and Jirsa 2011) introduced a generalized description of such coordination dynamics, as a combination of intrinsic dynamics (coordination types) and a strong coupling between them. Showcasing that, coordination can also arise with discrete (i.e. non-rythmic) bi-manual movements. More compelling evidence of the strong bi-manual coupling in humans can be seen when executing independent tasks. In the study of (Franz et al. 1991), the subjects were asked to perform two *independent* discrete movements with each hand, drawing a circle and a vertical straight line, respectively. The resulting shapes were *vertical* ellipses, elucidating the strong *ego-centric* constraints in bi-manual movements.

Humans also display an underlying coordination with external agents, for example when manipulating objects. Tasks such as lifting, carrying and reaching for large or heavy objects rely on bi-manual reaching behaviors which require not only *spatial* but also *temporal* constraints (Coats and Wann 2012). In a bi-manual reach, each hand has to adjust to the orientation, shape and size of the object while reaching for it. Moreover, the action of grabbing the object (i.e. closing the hands on the object) must be timed prior to rather than as a reaction to intercepting the object. Hence, bi-manual reaching requires to solve simultaneously *spatial* and *temporal* coordination constraints to move toward the object in coordination and to intercept the object (Vernon et al. 2011). Furthermore, when humans reach to different targets, their behavior might not exhibit spatial constraints, but the *temporal* constraints are still enforced. They smoothly adapt their respective speed to reach the targets simultaneously. Meaning that, there is an apparent timing synchronization even in asynchronous behaviors.

In our previous works (Salehian et al. 2016a, 2017), we offered a dynamical system (DS) based controller for *coordinated* multi-arm motion planning. The approach consists of a *virtual object*-DS control law that generates autonomous and synchronized motions for a multi-arm robot system. We use the notion of a *virtual object* to both *coordinate* the motion of the multiple robots with each other and with a moving object, such that the robots reach the dynamic object in *synchrony* (Figure 1a). Such a dynamical system emulates the strongly coupled human coordination strategies exhibited in the previously mentioned studies, providing predictable motions for humans.

In this paper, we improve upon our previous work, by tackling two main challenges that were not addressed in (Salehian et al. 2016a). First, we extend the *virtual object*-DS, such that it can generate two types of behaviors: (i) multi-arm *asynchronous* task-space behaviors, where each robot has its own target or desired motion (Figure 1a) and (ii) multi-arm *synchronous* task-space behaviors, where the robots' task is to *coordinate* with each other to *simultaneously* reach-for a moving object (Figure 1a). To provide a smooth transitioning between these two behaviors, we introduce the notion of *synchronization allocation*. Given the motion of the object and the joint workspace of the multi-arm system, each arm is being continuously allocated to a desired behavior. While being allocated to the *synchronous* behavior, control of the robots is taken over by the *virtual*

object-DS. While allocated to the *asynchronous* behavior, the robots are controlled independently, each with their own goal-directed stable DS. The proposed *multi-arm*-DS is expressed as a Linear Parameter Varying (LPV) system subject to stability constraints, that ensure convergence to the object or targets. Second, to ensure self-collision avoidance at the joint-level, we propose a *centralized* IK solver, formulated as a constrained convex optimization problem subject to data-driven self-collision avoidance (SCA) constraints. These SCA constraints are introduced as linear inequality constraints in the optimization problem in the form of a continuous "SCA Boundary function" and its gradient. We then propose to efficiently encode this SCA boundary function through a sparse kernel Support Vector Machine (SVM) (Joachims and Yu 2009), learned *a priori* from a simulated dataset of feasible configurations of the multi-arm system. The contributions of this paper, compared to our previous work (Salehian et al. 2016a), are thus threefold:

1. Unification of *synchronous* and *asynchronous* multi-arm behaviors in a single dynamical system.
2. A Self-Collision Avoidance (SCA) IK solver formulated as a convex QP problem that can be solved in real-time.
3. A data-driven learning approach to model the SCA constraints in a computationally efficient way.

This paper is organized as follows. Section 2 highlights related work on multi-arm control and SCA strategies. In Section 3, we present the *multi-arm*-DS, including a formalization and convergence proof of the LPV DS. Section 4 introduces a centralized inverse kinematic solver which can handle SCA constraints. In Section 5, we present the learning approach to approximate the safe manipulation regions fed to the SCA-IK solver. The proposed method is then validated with a dual-arm platform for several coordination and reaching scenarios in Section 6. Discussions and future work are presented in Section 7.

2 Related Work

Coordinating multiple robotic arms for object manipulation has been extensively studied in the robotics literature, one can find comprehensive surveys on dual/multi-arm motion planning in Wimböck et al. (2012); Wimböck and Ott (2012); Smith et al. (2012); Caccavale and Uchiyama (2016). However, the problem of planning the reach to grasp motion for a moving object with a multi-arm robotic system, while keeping coordination constraints, is a new field of research. The sole work that tackles a similar problem is that of Vahrenkamp et al. (2012, 2010), who proposed an Rapidly-exploring Random Tree (RRT)-based algorithm to generate collision free motions to grasp an object at rest with a bi-manual platform. Given its search-based strategy, this approach can guarantee feasible grasps by both arms and self-collision avoidance. However, due to its computational complexity and the fact that it cannot guarantee simultaneous interception of the object by all arms, it becomes inadequate when trying to reach for a *moving* object. (Chung and Slotine 2009) proposed a contraction based control algorithm for synchronizing multi robotic arms with an external agent. However, the kinematic feasibility of the intercept point and the dual-behavior scenarios have not been addressed. To the best of the author's knowledge, there is no related work in the field of *reaching* for large *moving* objects with multi-arm systems, with the capabilities of *smoothly transitioning* to *asynchronous* behaviors, all the while avoiding for *self-collision* at the joint-level. Yet, for the sake of completeness, we briefly summarize works from the vast literature in multi-arm control that are relevant to each of our contributions.

2.1 De-Centralized Multi-Arm Control Strategies

In de-centralized control architectures, the robots are controlled separately by their own local controllers (Liu and Arimoto 1998). In early approaches, the coordination between a dual-arm system is achieved by categorizing them into two categories; namely a master and a slave. The motion of the master robot is assumed known whereas the slave robot must follow the master's motion while satisfying the closed-chain geometrical constraints (Luh and Zheng 1987). Similarly, (Gams et al. 2015) proposed a control architecture to perform a task of lifting an unknown object with a dual-arm system, where the slave arm is synchronized with the master arm through a coupling guided by position and velocity feedback errors. Although computationally efficient, this strategy assumes a fixed master-slave relationship, which, when dealing with moving objects, may adversely affect performance if the arms need to switch responsibility to perform the task on-line. In (Bai and Wen 2010), by using a velocity feedback and force feed-forward strategy, a de-centralized controller is proposed for transporting a flexible payload at a constants speed with multiple arms. In this approach master/slave roles are not assigned. However, as it assumes that each robot is a point-mass system it could not readily be applied to manipulating a moving object with unpredictable dynamics, as considered in our approach.

2.2 Centralized Multi-Arm Control Strategies

Some of the shortcomings of the de-centralized control architectures can be addressed by centralized control strategies (Aghili 2013; Suda et al. 2003; Wang et al. 2015). These strategies consider the robots and the manipulated object as a closed kinematic chain. In this line, an impedance control architecture for dual-arm manipulation is proposed in (Wimböck and Ott 2012), where the two end-effectors and a virtual frame, which is a function of the end-effectors' poses, are coupled via spatial springs. (Zhu 2005) proposed a motion synchronization controller to coordinate the end-effectors of dual-arm system when they are rigidly or flexibly holding a massless object. In (Likar et al. 2013), the motions of two robotic arms are synchronized by a velocity level motion synchronization algorithm. By concatenating the kinematic of two arms and the object, they introduced an augmented kinematic chain. The corresponding Jacobian is calculated to control the augmented kinematic chain by solving the inverse kinematics problem at the velocity level.

By exploiting advantages of centralized and de-centralized impedance control strategies, (Caccavale et al. 2008) proposed a control architecture to achieve a desired impedance at both the object and the end-effector levels. Similarly, (Chiacchio and Chiaverini 1998) proposed a two

level control architecture. Initially, the desired task variables are transformed into the corresponding joint-space motions by solving a centralized inverse kinematic problem. Then, the desired joint motions are fed to a decentralized joint-space controller.

All previously mentioned works assume that the object is firmly attached to the robots and modeled via a virtual object frame or by closing the kinematic chain. In this work, we leverage the idea of the *virtual object* to address the problem of coordination. The term *virtual object* is mostly used in robotics literature to represent the internal forces of a grasping task (Williams and Khatib 1993; Wimbock et al. 2008). In this paper, however, this term is used to achieve coordination at two levels. In the first level, the motion of each robot is coordinated with all robotic arms. In the second level, the resultant motion of the arms are coordinated with that of the object to satisfy the coordination constraints (i.e. for *synchronous* behavior).

2.3 Multi-Arm Self-Collision Avoidance

Self-collision avoidance is one of the main challenges in multi-arm manipulation. It is particularly relevant in the humanoid robot community and hence, has been extensively studied. Throughout the years, the approaches for solving collision avoidance for manipulation or locomotion in humanoids can be categorized into two types: (i) *planning* methods which generate feasible collision-free trajectories of known/quasi-static environments (Gharbi et al. 2009; Vahrenkamp et al. 2012; Escande et al. 2007; Vahrenkamp et al. 2010; Escande et al. 2014; Chrètien et al. 2016) and (ii) *reactive* approaches which solve collision-avoidance through the IK problem online (Ge and Cui 2000; Santis et al. 2007; Sugiura et al. 2007; Fang et al. 2015). For a comprehensive review on collision avoidance strategies for bi-manual systems refer to Petrič et al. (2015).

The main disadvantage of (i) planning approaches is their computational cost. Although much progress has been achieved in this regard, most planning approaches are still restricted to static scenarios. For example, in a dynamic scenario, where the robots must adapt to fast external perturbations, solving for collision avoidance must be in a matter of $1-2ms$. Typical computation times for *efficient* solvers in humanoid scenarios, lie between $\approx 700ms \rightarrow \approx 1s$ (Kanehiro et al. 2012; Orthey and Stasse 2013). Recently, (Chrètien et al. 2016) showed that the computation time of one iteration (for their collision-avoidance humanoid trajectory solver) is $2715ms$ on a single thread CPU. They prove, however, that with the use of parallel computing, their computation time can be improved significantly; i.e. to $54ms$ on a GPU. Though significant, this improvement is far from the requirements of dynamic scenarios. The sole approach which has proven to achieve dynamic adaptation at $< 1ms$ is that of (Murray et al. 2016), where an FPGA robot-specific chip is designed for dynamic motion planning. Although promising, this approach is limitated by the need of computing a probabilistic roadmap *a priori* and its dependence on a specific robot configuration.

Reactive approaches (ii), on the other hand, are not hindered by computational inefficiency. In the works of (Santis et al. 2007; Sugiura et al. 2007) repulsion forces are computed from colliding segments to generate self-collision avoidance motions. (Fang et al. 2015) proposed a hierarchical-based algorithm to find in-danger points and solve the IK problem such that the distance between these points is maximized. These approaches, although fast, suffer from the same shortcoming as generic potential-field obstacle avoidance algorithms; i.e. the stability of the generated motion is not guaranteed, as the robot might get stuck at local minima. Moreover, potential-field approaches suffer from providing no passages between closely spaced obstacles and the possibility of oscillations in presence of obstacles (Ge and Cui 2000).

All of the previously mentioned works, be it (i) *planning-based* or (ii) *reactive*, have a common methodology: they rely on computing *minimum distances* between links/joints/segments/objects (represented as sphere/swept-spheres/polygons) to detect/avoid collisions. The use of *minimum distances* for collision avoidance inherently introduces non-linear and non-convex constraints to an otherwise convex optimization problem (Ratliff et al. 2015). This, in fact, is the main reason (i) *planning* algorithms rely on *computationally inefficient* global optimization or trajectory optimization methods and that (ii) *reactive* methods tend to get stuck in local minima. To this end, approaches based on signed distance fields have been successful in encoding proximity of obstacles as continuous costs in local trajectory optimization frameworks; by either providing explicit cost gradients (Ratliff et al. 2009; Zucker et al. 2013) or through derivative-free stochastic optimization methods (Kalakrishnan et al. 2011). Such approaches, however, fail to recover when solving for optimization problems that become ill-defined due to particular shapes of obstacles or many local minima. To alleviate this, (Ratliff et al. 2015) provide a general motion optimization framework which exploits the Riemannian geometry of the workspace to represent costs for obstacle avoidance, kinematic limits, etc, by warping the workspace via Riemannian metrics and their gradients. These approaches, although promising, are still limited by their computational efficiency, as they all present optimization times of $\approx 0.5s$.

In this work, we focus on solving the self-collision avoidance problem *efficiently*; i.e. in $< 2ms$. We work-around the limitations of the previously mentioned approaches, by learning a continuous and continuously differentiable function $\Gamma(.)$ from a dataset of "collided" and "non-collided" multi-arm configurations. $\Gamma(.)$ represents the region of feasible and infeasible robot configurations, implicitly encoding a distance in *feature space*, of the current robot configuration to a "collided" configuration. By formulating $\Gamma(.)$ as the prediction rule of a kernel Support Vector Machine (SVM) (Scholkopf and Smola 2001) we can compute a continuous $\nabla\Gamma(.)$ on-line. We hence propose a centralized IK solver as a convex QP optimization problem subject to *linear* inequality constraints imposed by $\Gamma(.)$ and $\nabla\Gamma(.)$, avoiding (i) the computation of $\min(||.||_2)$ pair-wise distances between joints/links/segments and (ii) the need for trajectory optimization. Through the use of a sparse Support Vector Machine approximation (Joachims and Yu 2009) we learn an efficient representation of $\Gamma(.)$ that enables us to solve the QP optimization problem in less than $2ms$, implemented on a single threaded CPU.

3 Coordinated Multi-Arm Motion Planning

In order to achieve the envisioned scenario; i.e. smoothly transition between *asynchronous* and *synchronous* behaviors depending on the object's motion, two main challenges should be addressed:

1. Prediction of the object's trajectory and computation of its feasible intercept points.
2. Planning stable multi-arm motions towards their corresponding object intercept points (when allocated to *synchronous* behavior), or individual targets (when allocated to *asynchronous* behavior) .

The proposed solutions to these challenges are described in sub-sections 3.1 and 3.2, respectively. Once the multiple end-effector motions are generated, these are mapped to desired joint-space configurations by a centralized IK solver which uses a learned model of collision-free regions and the robots' kinematic constraints, this is described in Sections 4 and 5. For simplicity and practicality, we summarize the most relevant notations used throughout the paper in Table 2 and illustrate them in Fig.2. The control flow of the entire framework is illustrated in Fig. 22.

3.1 Object Trajectory and Intercept-Point Prediction

Once the *main* object starts moving towards the robots, a linear model predicts its progress ahead of time and determines a point along its trajectory where the object will become reachable by all robotic arms. We do not assume a known model of the dynamics of the object. The sole knowledge about the object is the location of its *reaching points*. These correspond to user-defined position and orientation of the arms at the reaching point (see Fig. 2).

To find the *feasible intercept point*, which is the point where the object can be reached by all robots, at its predefined *reaching points*, we model the workspace of each robot via a probabilistic classification scheme. The reachable workspace of each robot, $p_j(\xi;\theta_j^W)\ \forall j \in \{1,\dots,N_R\}$ for N_R robots, is modeled through a Gaussian Mixture Model (Bishop 2007). If $p_j({}^j\xi_i^O) > \delta_j$, where δ_j is a minimum likelihood threshold, then ${}^j\xi_i^O$, i.e. the i-th reaching position on the object (sub-script) in the j-th robot's reference frame (left super-script), is classified as a *feasible* position for the j-th robot to reach. As the reachable workspaces of each robot are statistically independent from each other, we can calculate the joint distribution of all workspaces by computing the product of distributions, as follows:

$$p(\{{}^1\xi_1^O,\dots,{}^{N_R}\xi_{N_R}^O\};\Theta^W) = \prod_{j=1}^{N_R} p_j({}^j\xi_j^O;\theta_j^W) \qquad (1)$$

where $\Theta^W = \{\theta_1^W,\dots,\theta_{N_R}^W\}$ is the set of parameters for all robot workspaces and $\{{}^1\xi_1^O,\dots,{}^{N_R}\xi_{N_R}^O\}$ are the reaching positions in each robot's reference frame. The *minimum joint likelihood threshold is* $\delta = \prod_{j=1}^{N_R}\delta_j$. if $\exists T^* : \delta < p({}^1\xi_1^O(T^*),\dots,{}^{N_R}\xi_{N_R}^O(T^*);\Theta)$, the object at T^* $(\xi^O(T^*) = \frac{1}{N_R}\sum_{j=1}^{N_R}\xi_j^O(T^*))$ is classified as the *feasible intercept point*. If more than one point on the predicted trajectory is classified as the *feasible intercept point*, we select the closest one, in Euclidean space position, to the robot end-effectors. Refer to (Salehian et al. 2016a) for details of this procedure.

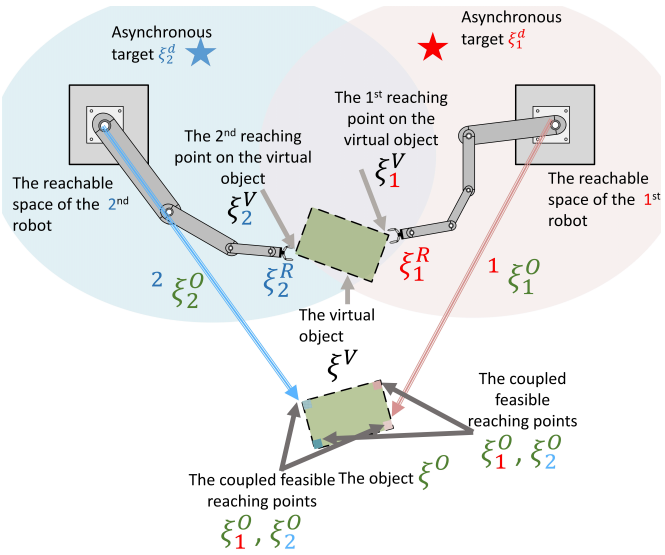


Figure 2. An illustration of the variables for $N_R = 2$. The reachable areas are feasible areas for grasping the object. Except for ${}^2\xi^O{}_2$ and ${}^1\xi^O{}_1$, the variables are expressed in the reference frame located on the desired intercept point; i.e. $\xi^O(T^*) = \begin{bmatrix}0 & \dots & 0\end{bmatrix}^T$.

3.2 Dynamical System-based Control of Multi-Arm Systems

Once the *feasible intercept point* is found, the motion of i-th robot's end-effector $\forall i \in \{1,\dots,N_R\}$, is generated by following a Linear Parameter Varying (LPV) dynamical system (DS), composed of both *synchronous* and *asynchronous* behaviors, and coupled to the motion of the object (ξ^O) through a *virtual object* (ξ^V) (see Figure 3) as follows:

$$\begin{aligned}\dot{x}_i^R = \dot{\tau}_{c_i}\left(x_i^V - x_i^d\right) + \tau_{c_i}\dot{x}_i^V + \\ \mathbf{A}_i\left(\theta_i(x_i^R)\right)\left(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d)\right),\end{aligned} \qquad (2)$$

where $x_i^R(t) = \begin{bmatrix}\xi_i^R & \dot{\xi}_i^R\end{bmatrix}^T \in \mathbb{R}^{2d_n}$ and $x_i^V(t) = \begin{bmatrix}\xi_i^V & \dot{\xi}_i^V\end{bmatrix}^T \in \mathbb{R}^{2d_n}$ are the state of the i^{th} end-effector and *virtual object*, respectively.[1] The state of the virtual object is used to guide the robots for *synchronous* behaviors. In the case of *asynchronous* behaviors, each i-th robot has its own static target, denoted as $x_i^d = \begin{bmatrix}\xi_i^d & 0\end{bmatrix} \in \mathbb{R}^{2d_n}$. $0 \le \tau_{c_i} \le 1$ is the *synchronization allocation* parameter and is of class $\mathscr{C}^1$. $\theta_i(x_i^R) \in \mathbb{R}^{d_{s_i}\times 1}\ \forall i \in \{1,\dots N_R\}$ is a vector of scheduling parameters; $\theta_i(x_i^R) = \begin{bmatrix}\theta_{i1}(x_i^R) & \dots & \theta_{id_{s_i}}(x_i^R)\end{bmatrix}^T$ and for simplicity of the notations, its argument is dropped in the rest of the paper. $\mathbf{A}_i(.) : \mathbb{R}^{d_{s_i}\times 1} \to \mathbb{R}^{2d_n\times 2d_n}$ are the affine dependence of state-space matrices on the scheduling parameter and the state vectors:

$$\mathbf{A}_i(\theta_i) = \sum_{k=1}^{d_{s_i}} \theta_{ik}A_{ik} \quad A_{ik} \in \mathbb{R}^{2d_n\times 2d_n} \quad \theta_{ik} \in \mathbb{R}^{1\times 1} \qquad (3)$$

The parameters of LPV systems can be approximated with regression models; i.e. polynomial, periodic functions or Gaussian Mixture Regression (GMR). In this paper, the latter approach is used to approximate the scheduling parameters

from kinematically feasible demonstrations. The advantage of this technique is that it inherently results in normalized scheduling parameters; i.e. $0 < \theta_{ik} \leq 1,\ \sum_{k=1}^{d_{s_i}} \theta_{ik} = 1,\ \forall i \in \{1,\ldots,N_R\}$, refer to (Salehian et al. 2016b) for further details on this approximation approach.

Theorem 1. *The dynamical systems given by* (2) *asymptotically converge to* $\tau_{c_i} x_i^V + (1-\tau_{c_i}) x_i^d$*; i.e*

$$\lim_{t\to\infty} \| x_i^R(t) - \tau_{c_i}(t) x_i^V(t) + (\tau_{c_i}(t) - 1) x_i^d \| = 0 \tag{4}$$

if there exist $P_i^R,\ Q_i^R$ *such that:*

$$\begin{cases} 0 \prec P_i^R \quad 0 \prec Q_i^R \\ P_i^R A_{ik} + A_{ik}{}^T P_i^R \prec -Q_i^R \quad \forall k \in \{1,\ldots,d_{s_i}\} \\ 0 \leq \theta_{ik} \end{cases} \tag{5}$$

Moreover, by taking $\dot{\tau}_{c_i}(x_i^V - x_i^d) + \tau_{c_i}\dot{x}_i^V - \mathbf{A}_i(\theta_i(x_i^R))(x_i^d + \tau_{c_i}(x_i^V - x_i^d))$ *as the input and* $x_i^R(t)$ *as the output of the dynamical system* (2)*,* (2) *is passive if* (5) *is satisfied.*

Proof: see Appendix B and C.

In (5), P_i^R and Q_i^R are auxiliary matrices which are used in a Lyapunov stability and convergence proof provided in Appendix B. It is important to remark that the Lyapunov function used in these stability proofs and hence, the matrices P_i^R and Q_i^R, are never evaluated, since it is merely their existence that is required. In (2), τ_{c_i} determines the level of *synchronization* between the i^{th} robot and the virtual object, see Fig. 3. Assuming that τ_{c_i} is constant and (5) is satisfied, when $\tau_{c_i} = 0$, (2) yields an *asynchronous* DS for reaching towards individual targets:

$$\dot{x}_i^R = \underbrace{\mathbf{A}_i(\theta_i)(x_i^R - x_i^d)}_{\text{Reaching individual target } (x_i^d)} \rightarrow \left\{ \lim_{t\to\infty} \|x_i^R - x_i^d\| = 0 \right. \tag{6}$$

Similarly, when $\tau_{c_i} = 1$, (2) results in a perfect tracking DS of the i^{th} reaching point on the virtual object:

$$\dot{x}_i^R = \underbrace{\dot{x}_i^V + \mathbf{A}_i(\theta_i)(x_i^R - x_i^V)}_{\substack{\text{Tracking } i^{\text{th}} \text{ reaching} \\ \text{position on the virtual object}}} \rightarrow \begin{cases} \lim_{t\to\infty} \|x_i^R - x_i^V\| = 0 \\ \lim_{t\to\infty} \|\dot{x}_i^R - \dot{x}_i^V\| = 0 \end{cases} \tag{7}$$

To *smoothly transition* between these behaviors, one could calculate $\tau_{c_i}\ \forall i \in \{1,\ldots,N_R\}$ with a continuous logistic function. In this paper, however, we propose the following DS which varies $\tau_{c_i}\ \forall i \in \{1,\ldots,N_R\}$ such that $\tau_{c_i} \to 1$ when the object moves towards the robots and $\tau_{c_i} \to 0$ when it moves away:

$$\begin{aligned} \dot{\tau}_{c_i}(t) &= \frac{\tau_{c_i}(1-\tau_{c_i}) G(\xi^O(t), \dot{\xi}^O(t))}{\mathbf{k}} \\ \tau_{c_i}(0) &= \varepsilon \quad ,\forall i \in \{1,\ldots,N_R\} \\ G(.) &= -\frac{\dot{\xi}^O(t)^T (\xi^O(t) - \xi^O(T^*))}{\varepsilon + \|\xi^O(t) - \xi^O(T^*)\|^2} \end{aligned} \tag{8}$$

Where, $0 < \varepsilon \ll 1$ is a small positive value. $\mathbf{k} \in \mathbb{R}_{>0}$ is a positive constant that controls for the steepness of the increase or decrease of the parameter.[2] As the initial value of τ_{c_i} is positive < 1, (8) is a bounded dynamical system between $[0,1]$. $G(.)$ is a function that *coordinates* the robots with the *virtual object*, such that if the real object moves toward the workspaces, the robots perform the *synchronous* behavior, otherwise they fall back to the *asynchronous* behavior. The main advantage of the proposed criterion is its adaptability. $\text{sgn}(\tau_{c_i})$ changes with respect to the direction of the object's motion; when the object approaches the robots, $\text{sgn}(\tau_{c_i}) \to (+)$, otherwise, $\text{sgn}(\tau_{c_i}) \to (-)$. Consequently, if the object is moving towards the robots, they are *synchronized* with the virtual object. Otherwise, they perform the *asynchronous* behavior.

As the *synchronization allocation* parameters vary over time, the *virtual object*-DS proposed in (Salehian et al. 2016a), which generates the motion of the virtual object, is no longer applicable. To appropriately consider the effects of the *synchronization* parameters on the motion of the virtual object, and consequently of the robots, the following DS is proposed to generate the motion of the virtual object.

$$\dot{x}^V(t) = \frac{1}{1 + \sum_{i=1}^{N_R} \tau_{c_i}} \left(\gamma \dot{x}^O + \dot{\gamma} x^O + A^V \left(x^V - \gamma x^O \right) + \sum_{i=1}^{N_R} U_i \right) \tag{9}$$

Where, $x^V(t) = \begin{bmatrix} \xi^V(t) & \dot{\xi}^V(t) \end{bmatrix}$ is the state of the virtual object. $0 < \gamma < 1$ is the *coordination* parameter and is of class $\mathscr{C}^1$. U_i is the interaction effect of the motion of the i^{th} end-effector on the virtual object, based on (2) and (9):

$$\begin{aligned} U_i = {} & \dot{x}_i^R - \mathbf{A}_i(\theta_i) \left(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d) \right) \\ & - \dot{\tau}_{c_i} \left(x_i^V - x_i^d \right) \end{aligned} \tag{10}$$

By substituting, (2) and (10) into (9), we have:

$$\dot{x}^V(t) = \frac{1}{1 + \sum_{i=1}^{N_R} \tau_{c_i}} \left(\gamma \dot{x}^O + \dot{\gamma} x^O + A^V (x^V - \gamma x^O) + \sum_{i=1}^{N_R} (\tau_{c_i} \dot{x}_i^V) \right) \tag{11}$$

Theorem 2. *The dynamical system given by* (11) *asymptotically converges to* $\begin{bmatrix} \gamma(t)\xi^O(t) & \gamma(t)\dot{\xi}^O(t) + \dot{\gamma}(t)\xi^O(t) \end{bmatrix}^T$ *i.e.*

$$\begin{aligned} &\lim_{t\to\infty} \|\xi^V(t) - \gamma(t)\xi^O(t)\| = 0 \\ &\lim_{t\to\infty} \|\dot{\xi}^V(t) - (\gamma(t)\dot{\xi}^O(t) + \dot{\gamma}(t)\xi^O(t))\| = 0 \end{aligned} \tag{12}$$

if there exit $P_i^V,\ Q_i^V$ *such that:*

$$\begin{cases} 0 \prec P^V \\ 0 \prec Q^V \\ P^V A^V + A^V{}^T P^V \prec -Q^V \end{cases} \tag{13}$$

Moreover, by taking $\gamma\dot{x}^O + \dot{\gamma}x^O - A^V(\gamma x^O)$ *as the input and* x^V *as the output of the dynamical system* (11)*,* (11) *is passive if* (13) *is satisfied.*

Proof: : see Appendix D, E.

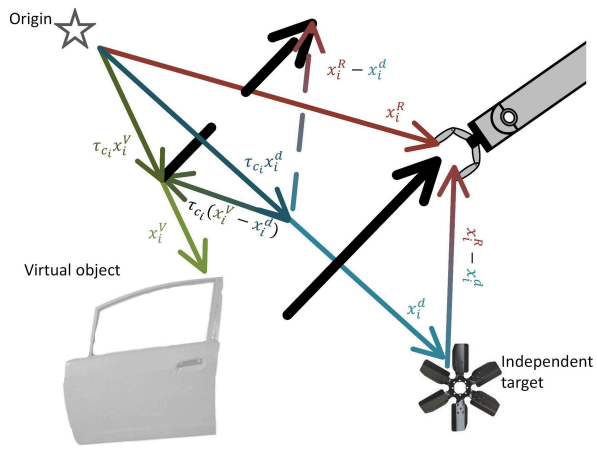


Figure 3. An illustration of the variables in (2). The colors of the variables and the arrows are corresponding; i.e. red represents the arm , green represents the virtual object and blue represents the independent target. The black arrows illustrates $(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))$. The dashed lines are used to show how the resultant vector is calculated.

Remark 1. *if* $\tau_{c_i} = 1$ *, based on* (7) *and* (12)*, the robots asymptotically converge to the reaching points on the object; i.e:*

$$\begin{aligned} &\lim_{t\to\infty} \|\xi_i^R(t) - \gamma(t)\xi_i^O(t)\| = 0 \\ &\lim_{t\to\infty} \|\dot{\xi}_i^R(t) - (\gamma(t)\dot{\xi}_i^O(t) + \dot{\gamma}_i(t)\xi_i^O(t))\| = 0 \end{aligned} \quad \text{if } \tau_{c_i} = 1 \quad (14)$$

In (13), P_i^V and Q_i^V are auxiliary matrices which are used in the stability and convergence proof; see Appendix D. If $\tau_{c_i} = 1$ and $(\gamma(t) = \dot{\gamma}(t) = 0)$, (11) generates asymptotically stable motions towards the predicted intercept point: i.e. coordination between the robots is preserved, but the coordination between the robots and the object is lost. If $\tau_{c_i} = 1$ and $(\gamma(t) = 1, \dot{\gamma}(t) = 0)$, (11) generates asymptotically stable motions towards the real object, even though its motion is not accurately predicted: i.e. perfect *coordination* with the object.[3] However, in this case, there is no guarantee that the virtual object intercepts the real object inside the workspace of the robots, i.e. because of the robots' kinematic constraints, the coordination between them is lost. Thus, one can vary the values of the coordination parameters between $[0,1]$, such that $\gamma = 1$ at the vicinity of the desired intercept time as proposed in (Salehian et al. 2016a):

$$\dot{\gamma} = \frac{1-\gamma}{\|\xi^O(t) - \xi^O(T^*)\| + \varepsilon} = \frac{1-\gamma}{\|\xi^O(t)\| + \varepsilon}, \quad \gamma(0) = 0. \quad (15)$$

(15) improves the robustness of the multi-arm reaching motion in face of inaccuracies in the object's motion prediction, as it ensures that when the object is close enough to the *feasible reaching positions*, the virtual object converges to the real object and perfectly tracks it; i.e. $\gamma(T^*) = 1$. Hence, the robots can simultaneously track the desired reaching points on the object in coordination. The C++ implementation of this approach is provided in *Multiarm_ds*(Table 3). The proposed algorithm can only guarantee collision-avoidance between end-effectors, via the *virtual object*, for *synchronous* behaviors. In the following section we present a centralized inverse kinematics solver, that addresses self-collision avoidance at all times.

4 Self-Collision Avoidance (SCA)

To avoid collisions between the joints of the arms, we need to devise a control algorithm to ensure that none of the robots' body parts collide with each other. To achieve this objective, the IK solver must consider not only the kinematic constraints of each robot, but also self-collision constraints. Given that the robots' bases are fixed wrt. each other, we can explore the joint workspace of the robots, in order to model the regions that may lead to collision. Since the space of joint configurations is continuous, we must approximate the regions of collisions by building a continuous map of the feasible (safe) and infeasible (collided) configurations. Assuming that the infeasible regions can be bounded through a continuous and continuously differentiable function $\Gamma(q^{ij}) : \mathbb{R}^{d_{q_i}+d_{q_j}} \to \mathbb{R}$, where $q^{ij} = [q^i, q^j]^T \in \mathbb{R}^{d_{q_i}+d_{q_j}}$ are the joint angles of the i^{th} and j^{th} robots, respectively. We define $\Gamma(.)$ such that:

$$\begin{aligned} &\text{Collided configurations:} && \Gamma(q^{ij}) < 1 \\ &\text{Boundary configurations:} && \Gamma(q^{ij}) = 1 \\ &\text{Free configurations:} && \Gamma(q^{ij}) > 1 \end{aligned} \quad (16)$$

A data-driven approach for building $\Gamma(q^{ij})$ is proposed in Section 5. Fig. 4 illustrates a $\Gamma(.)$ for a toy $2D$ example. (16) provides constraints that must be taken into account when solving the inverse kinematics (IK) problem. We propose the following quadratic program to solve the IK:

$$\underbrace{\underset{\dot{\mathbf{q}}}{\text{argmin}} \frac{\dot{\mathbf{q}}^T W \dot{\mathbf{q}}}{2}}_{\text{Minimize expenditure}} \quad (17a)$$

Subject to:

$$\underbrace{\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \dot{\xi}^R}_{\text{Satisfy the desired end-effector motion}} \quad (17b)$$

$$\underbrace{\dot{\theta}^- \leq \dot{\mathbf{q}} \leq \dot{\theta}^+}_{\text{Satisfy the kinematic constraints}} \quad (17c)$$

$$\underbrace{\begin{aligned} &-\nabla\Gamma^{ij}(q^{ij})^T \dot{q}^{ij} \leq \log(\Gamma^{ij}(q^{ij}) - 1) \\ &\forall (i,j) \in \{(1,2),(1,3),\ldots,(N_R-1,N_R)\} \end{aligned}}_{\text{Do not penetrate the collision boundary}} \quad (17d)$$

Where, $\mathbf{q} = [q^1, \ldots, q^{N_R}]^T \in \mathbb{R}^{d_\mathbf{q}}$, $d_\mathbf{q} = \sum_{i=1}^{N_R} d_{q_i}$.[4] W is a block diagonal matrix of positive definite matrices. $\mathbf{J} = diag(J_1, \ldots, J_{N_R})$ is block diagonal matrix of the Jacobian matrices. $\dot{\xi}^R = \begin{bmatrix} \dot{\xi}_1^R & \ldots & \dot{\xi}_{N_R}^R \end{bmatrix}^T \in \mathbb{R}^{d_\mathbf{n}}$, $d_\mathbf{n} = N_R d_n$ is the desired velocity given by (2). $\dot{\theta}^i = \begin{bmatrix} \dot{\theta}_1^i & \ldots & \dot{\theta}_{N_R}^i \end{bmatrix} \quad \forall i \in \{-,+\}$ and $\dot{\theta}_i^+ \in \mathbb{R}^m$ and $\dot{\theta}_i^- \in \mathbb{R}^m$ are conservative lower and upper bounds of the joint limits, respectively. To integrate the joint limits into the velocity level constraints, (Xia and Wang 2000) propose the following equation.

$$\begin{aligned} \dot{\theta}_i^- &:= \max\left(\mu(q_i^- - q_i), \dot{q}_i^-\right) \\ \dot{\theta}_i^+ &:= \min\left(\mu(q_i^+ - q_i), \dot{q}_i^+\right) \end{aligned} \quad (18)$$

With q_i^-, q_i^+,q_i^-, $\dot{q}^-$ as the conservative lower and upper bounds on the joints' positions and velocities. The *intensify coefficients*, $0 < \mu_p$, determine the magnitude of decelerations. This should be defined such that the feasible region of $\dot{\theta}$ is greater than the joint velocity limits.

While the robots are far from the boundary configurations, the value of $\log(\Gamma^{ij}(q^{ij})-1)$ is positive which relaxes the inequality constraints; i.e. the robots accurately follow the desired end-effector trajectory. When they are near the boundary configurations, the value of $\log(\Gamma^{ij}(q^{ij})-1)$ is negative. Therefore, constraint (17d) forces the joint angles to move away from the boundary as they approach it. Since satisfying the collision avoidance and the kinematic constraints is of higher priority than following the desired end-effector motion, we give higher penalty to (17c) and (17d), than to (17b). In a particular case, when the robots are initiated inside of the boundary (i.e. $\Gamma^{ij}(.) < 1$), $\log(\Gamma^{ij}(.)-1)$ is not defined. In this case, we replace $\log(\Gamma(.)-1)$ with a large negative number which pushes the robots outward towards the boundary.

Equation (17) is a convex quadratic programming (QP) problem with equality and inequality constraints, hence, there is no closed form solution for it. As the solutions to such linear optimization are solver-dependent, in terms of computation cost, we compare three approaches to solve (17). The *first* approach formulates (17) as a system of piecewise-linear equations and uses a DS-based approach to solve them (Xia and Wang 2000; Zhang et al. 2004; Zhang 2005). The *second* approach uses Nlopt, a standard nonlinear programming solver (Johnson 2016). The *third* approach uses a solver specifically designed for constrained convex problems; the solver which we use is called CVXGEN, introduced in (Mattingley and Boyd 2012), which generates C codes, tailored for the specific formulation of (17). As the *second* and *third* approaches are ready to use interfaces, in the rest of this chapter we introduce the *first* approach.[5]

Lemma 1. *Linear quadratic programming* (17) *is equivalent to the following system of piecewise-linear equations.*

$$P_\Omega(u-(Mu+b))-u=0 \tag{19}$$

Moreover, the following dynamical system is asymptotically stable to u^*, where $u^* \in \mathbb{R}^{d_u}$ is the solution of (19) .

$$\dot{u} = (I+M^T)(P_\Omega(u-(Mu+b))-u) \tag{20}$$

where

$$M = \begin{bmatrix} W & -\boldsymbol{J}(\boldsymbol{q})^T & -\nabla\Gamma(\boldsymbol{q}) \\ \boldsymbol{J}(\boldsymbol{q}) & 0 & 0 \\ \nabla\Gamma(\boldsymbol{q})^T & 0 & 0 \end{bmatrix} \tag{21a}$$

$$b = \begin{bmatrix} 0 \\ -\dot{\xi}^R \\ -\log(\Gamma(\boldsymbol{q})-1) \end{bmatrix}. \tag{21b}$$

$u = \begin{bmatrix} \dot{\boldsymbol{q}} & \eta & \nu \end{bmatrix}^T \in \mathbb{R}^{d_u}$; $d_u = d_{\mathbf{q}} + d_{\mathbf{n}} + 1$. *As $0 \preceq W$, M is also positive semi-definite. $\eta \in \mathbb{R}^{d_N}$ and $\nu \in \mathbb{R}$ are the dual decision vectors.* $P_\Omega(h) = \begin{bmatrix} P_\Omega(h_1) & \dots & P_\Omega(h_{d_u}) \end{bmatrix}$ *is the element-wise $\Omega-$ projection operator defined as*

$$P_\Omega(h_i) = \begin{cases} u_i^- & h_i < u_i^- \\ h_i & u_i^- \le h_i \le u_i^+ \\ u_i^+ & u_i^+ < h_i \end{cases} \quad \forall i \in \{1,\dots,d_u\}. \tag{22}$$

u^+ and u^- are the bounds of the primal-dual decision vector u defined as

$$u^- = \begin{bmatrix} \dot{\theta}^- \\ -\infty \\ 0 \end{bmatrix} \qquad u^+ = \begin{bmatrix} \dot{\theta}^+ \\ +\infty \\ +\infty \end{bmatrix} \tag{23}$$

and $\Omega = \{u \in \mathbb{R}^{d_u} | u^- \le u \le u^+\}$.

Proof: Refer to (Zhang 2005) and (Xia and Wang 2000).

Theorem 3. *By taking u and $(I+M^T)P_\Omega(u-(Mu+b))$ as the output and the input of the system* (20)*, respectively,* (20) *is passive.*

Proof: See Appendix F.

Remark 2. *Theorems 1, 2 and 3 show that all the proposed dynamical systems are passive. Hence, if the robots and the low level torque controllers are passive, the proposed framework for coordinated Multi-Arm system is passive and stable as it is a feedback system of passive elements.*

The C++ implementation of the proposed centralized inverse kinematic solver is provided by authors in *QP_IK_solver*, see Table 3 .

5 Learning a Self-Collision Avoidance (SCA) boundary

In this section, we introduce a data-driven approach to approximate the self-collision avoidance (SCA) boundary function $\Gamma(q^{ij})$, used in (17) as a constraint for the IK solver. As per (16), $\Gamma(q^{ij})$ should be of class $\mathscr{C}^0$ and $\mathscr{C}^1$ and, interestingly, can be formulated as a binary classification problem, $y \leftarrow \text{sgn}\left(\Gamma(q^{ij})\right)$ for $y \in \{+1,-1\}$, where "collided" joint configurations belong to the negative class (i.e. $y=-1$) and "non-collided" configurations belong to the positive class (i.e. $y=+1$). When $\Gamma(q^{ij})=1$, q^{ij} is at the boundary of the positive class; i.e. the self-collision boundary (Figure 4).

To recall, the configuration space of a multi-arm system is an n-dimensional torus ($S^1 \times S^1 \times \cdots \times S^1 (n-times) = \mathscr{T}^N$) for n DOF (LaValle 2006). Given two robots with 7DOF each, the manifold in which the multi-arm joint-angle vector lies in is $q^{ij} \in \mathscr{T}^{14}$. Employing q^{ij} as the feature vector for a classification problem can be problematic for several reasons. Firstly, many machine learning algorithms rely on computing distances/norms in Euclidean space, assuming the features are i.i.d. from an underlying distribution in $\mathbb{R}^N$. Hence, a Euclidean norm applied on $q^{ij} \in \mathscr{T}^N$, is merely an approximation of the actual distance in the $\mathscr{T}^N$ manifold. In fact, a proper distance metric for joint-angles, i.e. $d(q_1^{ij}, q_2^{ij})$ where $q^{ij} \in \mathscr{T}^N$, is non-existent. For this reason, most trajectory optimization planning algorithms rely on mapping joint-space configurations to task-space via

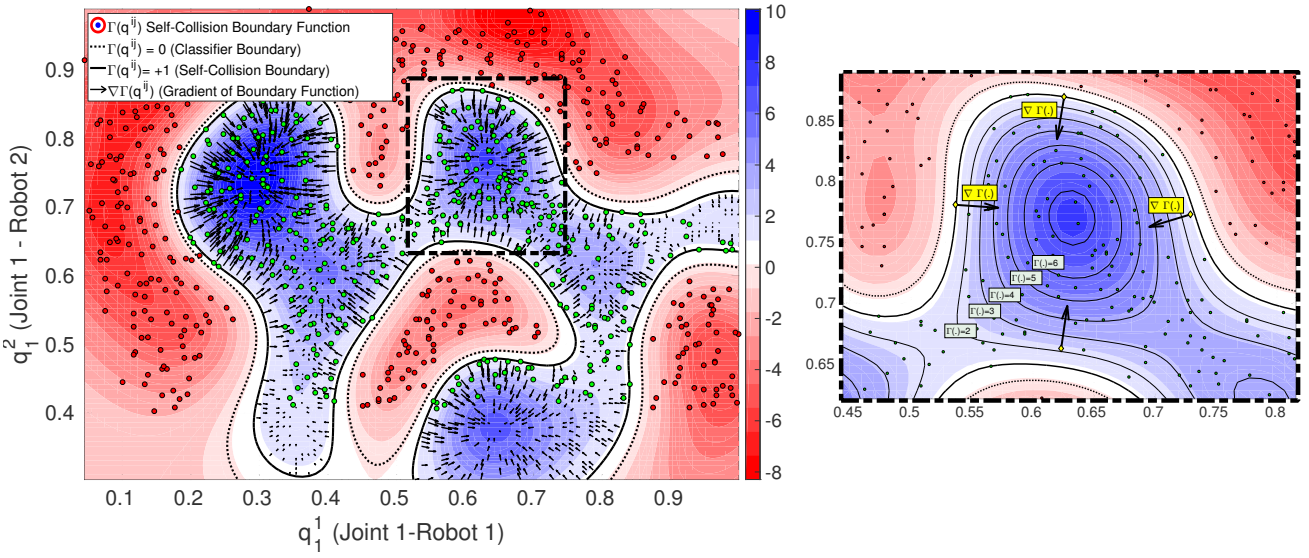


Figure 4. $\Gamma(q^{ij})$ function for a toy 2D example. Assume two robots with 1-DOF each corresponding to each axis; i.e. $q^{ij} = [q_1^1, q_2^1]$. The green data-points represent "collision-free" robot configurations ($y = +1$), while the red data-points represent "collided" robot configurations ($y = -1$). The background colors represent the values of $\Gamma(q^{ij})$; refer to colorbar for exact values, where the blue area corresponds to collision-free robot configurations ($\Gamma(q^{ij}) > 1$), the red area to collided configurations ($\Gamma(q^{ij}) < 1$). The **Arrows** inside the *collision-free* region denote $\nabla\Gamma(q^{ij})$. We can see how $\nabla\Gamma(q^{ij})$ pushes the robot configurations away from the self-collision boundary.

forward kinematics (Stilman 2007; Holladay and Srinivasa 2016), where distances are well-established.

Nevertheless, in kernel methods such as Support Vector Machines (SVM) (Scholkopf and Smola 2001), one can overlook the topology of the input data and instead lift it onto a higher-dimensional feature-space $\Phi : \mathbb{R}^N \to \mathbb{R}^F$, where the *kernel trick*, $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ for $\mathbf{x} \in \mathbb{R}^N$, is employed to find a separating hyperplane in the F-dimensional feature space. Neural Networks (NN) can also alleviate the need for proper representations of the topology of the input-space, as they are capable of learning highly complex and non-linear decision boundaries via soft sigmoid functions (Rojas 1996). However, employing such methods directly on q^{ij} comes at a cost, as the trade-off between model complexity and classification error would be hindered [6].

For such reasons, and in-line with the trajectory optimization literature, instead of learning our self-collision avoidance (SCA) decision boundary function $\Gamma(.)$ on the joint-angle data q^{ij}, we learn $\Gamma(.)$ on the 3D Cartesian representation of the joint-angles $f(q^{ij})$. As illustrated in Figure 5, $f(q^{ij})$ is a vector composed of the 3D Cartesian positions of all joints for the i-th and j-th robot, computed via forward kinematics. The feature vector for a dual-arm robotic system is thus $f(q^{ij}) \in \mathbb{R}^{3d_{q_i}+3d_{q_j}}$. We posit that, by using $f(q^{ij})$ instead of q^{ij}, we can achieve a better trade-off between model complexity and error rate. Moreover, since the output of $\Gamma(.)$ is expected to be a scalar (16), no extra computation is necessary, as $\Gamma(q^{ij}) \equiv \Gamma(f(q^{ij}))$. The linear inequality constraints in (17d) require $\nabla\Gamma(.)$ to be $\mathscr{C}$, this can also be provided by either SVMs or NNs.

In this work, we favor the use of SVMs, for two main reasons: (i) Learning a SVM is a convex optimization problem; hence, we can always reach a *global optimum*, whereas NNs rely on heavy parameter tuning and multiple initializations in order to avoid *local minimum* solutions. (ii) SVMs yield sparser models than NNs for high-dimensional non-linear classification problems, leading to better runtimes at the prediction stage. In the following sub-sections, we describe how we learn $\Gamma(f(q^{ij}))$ through the SVM formulation from a simulated dataset of "collided" and "non-collided" robot configurations. We present our method for constructing such a dataset, motivate and propose to use a sparse SVM learning algorithm (Joachims and Yu 2009), to achieve runtime limitations imposed by the robot control loop, as discussed later on.

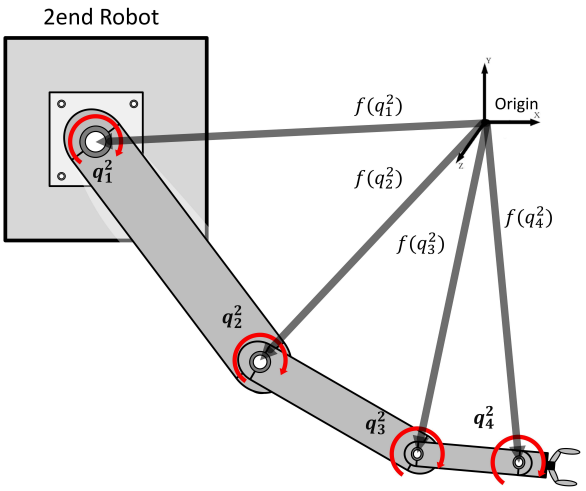


Figure 5. Illustration of $f(q^2)$, 3D Cartesian positions (e.g. $f(q_1^2) : S^1 \to \mathbb{R}^3$) of the individual joint angles in the multi-arm system reference frame used to learn the SCA Boundary function $\Gamma(f(q^2))$, where $f(q^2) = \{f(q_1^2), \ldots, f(q_2^4)\}$, assuming we have two robots with 4DOF each.

5.1 $\mathscr{C}^0$ and $\mathscr{C}^1$ Self-Collision Avoidance (SCA) Boundary via SVM

We follow the kernel Support Vector Machine (SVM) formulation and propose to encode $\Gamma\left(f(q^{ij})\right)$ as the SVM decision rule. By omitting the sign function and using the RBF Kernel $k\left(f(q^{ij}), f(q_n^{ij})\right) = e^{\left(-\frac{1}{2\sigma^2}||f(q^{ij})-f(q_n^{ij})||^2\right)}$, for a kernel width σ; $\Gamma\left(f(q^{ij})\right)$ has the following form,[7]

$$\begin{aligned}\Gamma\left(f(q^{ij})\right) &= \sum_{n=1}^{N_{sv}} \alpha_n y_n k\left(f(q^{ij}), f(q_n^{ij})\right) + b \\ &= \sum_{n=1}^{N_{sv}} \alpha_n y_n e^{\left(-\frac{1}{2\sigma^2}||f(q^{ij})-f(q_n^{ij})||^2\right)} + b,\end{aligned} \tag{24}$$

for N_{sv} support vectors, where $y_i \in \{-1,+1\}$ are the positive/negative labels corresponding to non-collided/collided configurations, $0 \leq \alpha_i \leq C$ are the weights for the support vectors which must yield $\sum_{n=1}^{N_{sv}} \alpha_n y_n = 0$ and $b \in \mathbb{R}$ is the bias for the decision rule. $C \in \mathbb{R}$ is a penalty factor used to trade-off between maximizing the margin and minimizing classification errors. Given C and σ, α_i's and b are estimated by solving the dual optimization problem for the *soft-margin* kernel SVM (Scholkopf and Smola 2001). Moreover, (24) naturally yields a continuous gradient as follows,

$$\begin{aligned}\nabla\Gamma\left(f(q^{ij})\right) &= \sum_{n=1}^{N_{sv}} \alpha_n y_n \frac{\partial k\left(f(q^{ij}), f(q_n^{ij})\right)}{\partial f(q^{ij})} \\ &= \sum_{n=1}^{N_{sv}} -\frac{1}{\sigma^2} \alpha_n y_n e^{\left(-\frac{1}{2\sigma^2}||f(q^{ij})-f(q_n^{ij})||^2\right)} \left(f(q^{ij}) - f(q_n^{ij})\right).\end{aligned} \tag{25}$$

Although $\nabla\Gamma(f(q^{ij}))$ already satisfies the constraints imposed by (17d), it lives in a $3d_{q_i}+3d_{q_j}$-dimensional space, $\nabla\Gamma\left(f(q^{ij})\right) \in \mathbb{R}^{3d_{q_i}+3d_{q_j}}$. We must then project this gradient onto its corresponding $\mathbb{R}^{d_{q_i}+d_{q_j}}$ joint-space; i.e. $\nabla\Gamma(q^{ij}) \in \mathbb{R}^{d_{q_i}+d_{q_j}}$ with the following expansion:

$$\nabla\Gamma\left(q^{ij}\right) = \frac{\partial\Gamma\left(f(q^{ij})\right)}{\partial f(q^{ij})} \times \frac{\partial f(q^{ij})}{\partial q^{ij}}, \tag{26}$$

the first term is equivalent to (25) and the second term is in fact the Jacobian of each 3D joint position wrt. each joint angle $J(q^{ij}) = \frac{\partial f(q^{ij})}{\partial q^{ij}}$ for which we have a closed-form solution.

5.2 Self-Collision Avoidance (SCA) Dataset Construction

In order to learn $\Gamma\left(f(q^{ij})\right)$, we must initially generate a dataset capable of identifying the so-called *self-collision boundary*. We begin by describing our simplified geometric representation of the robot's kinematic configuration used to identify "collided" and "non-collided" configurations.

For simplicity, let's assume a dual-arm setting, with each arm being a KUKA 7DOF (Figure 7). Similar to (Zucker et al. 2013), we simplify the representation of the robot's structure by fitting spheres to each joint and its adjoining physical structure as shown in Figure 6. By doing so, we

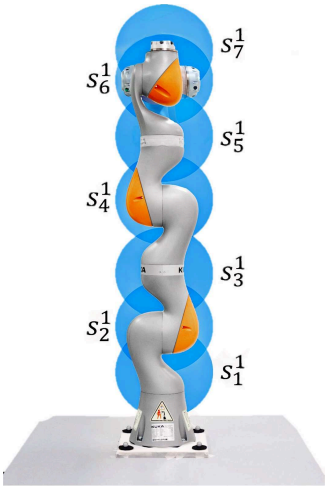


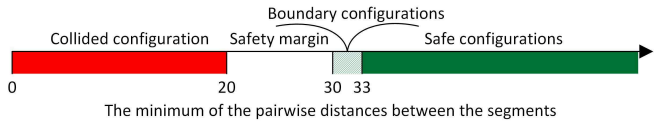
Figure 6. Illustration of $S^1 = \{s_1^1, \ldots, s_7^1\}$ for a KUKA IIWA 7DOF robot arm. S^i is the geometric representation of i-th robot with a set of spheres corresponding to each joint used to construct the SCA dataset. If a tool is attached to the arm, the last sphere is enlarged such that it encapsulates its extremities.

generate a discrete representation of the multi-arm robotic system as a set of spheres $S^{ij} = \{s_1^i, \ldots, s_7^i, s_1^j, \ldots, s_7^j\}$. By using spheres as a geometric representation of a joint, we simplify the distance computation between joints. As the distance from any point in a sphere to the nearest obstacle is lower-bounded to $d(c) - r$, where c is the center of the sphere and r its corresponding radius (Ratliff et al. 2009). Further, the lower-bound between two spheres is the distance between their centers (c_k^i) minus the sum of their respective radii (r_k^i), for the k-th spheres of the i-th robot. For example, given s_5^1 and s_7^2 the lower-bounded distance between them can be computed as $d(s_5^1, s_7^2) = d(c_5^1, c_7^2) - (r_5^1 + r_7^2)$.

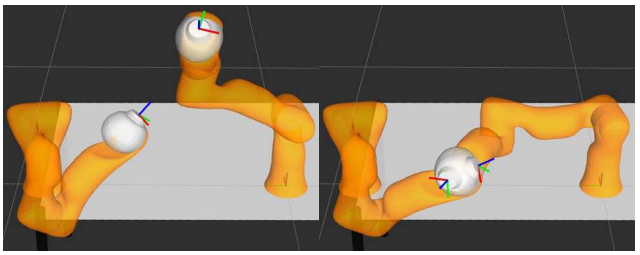
To identify collision in the dual-arm system, we compute the pairwise distances of the centers of the set of spheres of the i-th robot (S^i) wrt. the set of spheres of the j-th robot (S^j) and find the minimum distance $\min[d(c_{k^*}^1, c_{k^*}^2)]$. We then define a label for each robot configuration S^{ij} as follows,

$$y(S^{ij}) = \begin{cases} -1 & \text{if} \quad \min[d(c_{k^*}^1, c_{k^*}^2)] < (r_{k^*}^1 + r_{k^*}^2) \\ +1 & \text{if} \quad b_- \leq \min[d(c_{k^*}^1, c_{k^*}^2)] \leq b_+ \\ \emptyset & \text{if} \quad \min[d(c_{k^*}^1, c_{k^*}^2)] > b_+ \end{cases} \tag{27}$$

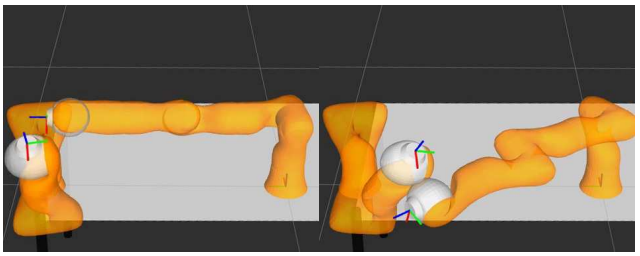
where $r_{k^*}^i$ corresponds to the radius of the k-th sphere, and b_-,b_+ correspond to minimum/maximum distances of the "safe" boundary. Specifically, a joint configuration is "collided", i.e. labeled as $y = -1$, when the $\min[d(c_{k^*}^1, c_{k^*}^2)]$ between the centers of the closest spheres is less than the sum of the radii of the corresponding spheres, i.e. $(r_{k^*}^1 + r_{k^*}^2)$. In practice, we set the spheres to a fixed radius of $10cm$, hence $(r_{k^*}^1 + r_{k^*}^2) = 20cm$. Given that virtually any robot configuration where $\min[d(c_{k^*}^1, c_{k^*}^2)] > (r_{k^*}^1 + r_{k^*}^2)$ can be considered "non-collided" configurations, we would end up with a heavily un-balanced dataset of "collided"/"non-collided" data-points. We thus, introduce a decomposition of the "non-collided" robot configurations into "boundary", labeled as $y = +1$, and "safe" configurations, which are not labeled $y = \emptyset$. If $\min[d(c_{k^*}^1, c_{k^*}^2)]$ lies within a safety margin, denoted by b_- and b_+, the robots are very close to each other but still safe, see Figure 7. We empirically found $b_- = 30cm$ and $b_+ = 33cm$ to be



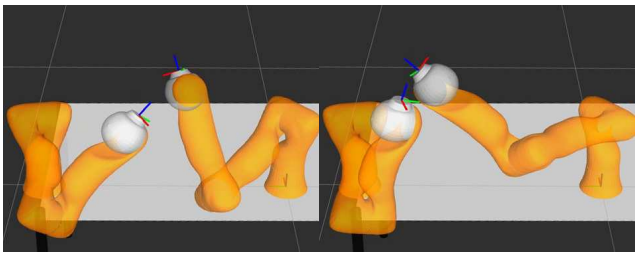
(a) Criteria for "collided", "boundary" and "safe" configurations.



(b) "boundary configuration" **(c)** "collided configuration"



(d) "boundary configuration" **(e)** "collided configuration"



(f) "boundary configuration" **(g)** "collided configuration"

Figure 7. Examples of the collided/boundary configurations of a *dual-arm setting* with an offset of $X_{off} = [0.0, 1.3.0.34]m$ between their bases (visualized in the RVIZ simulation environment (Rvi 2014)). (b), (d) and (f) are examples of the boundary configurations and (c), (e) and (g) are examples of the collided configurations.

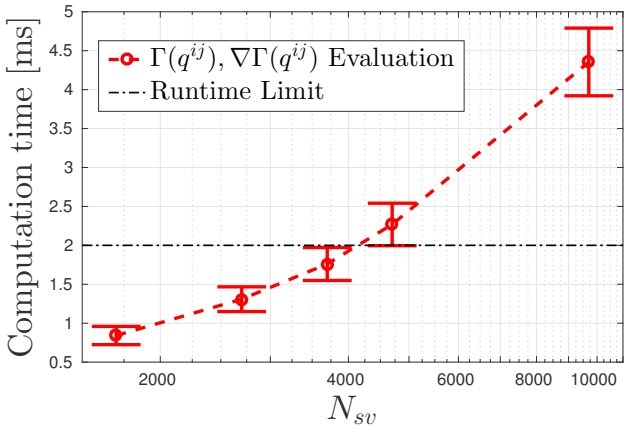


Figure 8. Comparison of runtime computational cost for evaluating $\Gamma(f(q^{ij}))$ and $\nabla\Gamma\left(f(q^{ij})\right)$ on a Dell Optiplex 3.4-GHz i7 PC with 8GB RAM for the *dual-arm setting* in Figure 7. The compared SVM models have the following increasing complexity $N_{sv} = \{1.7k, 2.7k, 3.7k, 4.7k, 9.7k\}$. The presented run-times are the mean and std. of $\approx 2k$ control loop cycles (equivalent to $\approx$ 4s), of the self-collision avoidance test. The maximum allowable N_{sv} in order to comply with the $2ms$ runtime limit is $N_{sv} \leq 3.7k$, which can only be obtained by using $\leq 1.5\%$ of this specific training dataset. Yet, the IK Solver takes $\approx 1ms$ to converge, hence, the actual limit would be around $N_{sv} \leq 3k$.

safe boundaries for our dual-arm setting. Hence, a "non-collided" configuration is in fact a "boundary" configuration, as all of the "safe" configurations are filtered out. This has a geometric meaning, rather than finding the margin between "collided" and "safe" configurations, our boundary function will model the tighter margin between "non-collided" and "boundary" configurations. From herein, we consider "boundary" configurations as the "non-collided" configurations.

To generate the positive $(y(S^{ij}) = +1)$ and negative samples $(y(S^{ij}) = -1)$ for our SCA dataset, we sample from all the possible motions of the robots in their respective workspaces and apply (27) to each configuration. To explore all possible joint configurations q^{ij}, we systematically displace all of the joints of both robots by 20 deg each. Joints q^i_1, q^i_3, q^i_5, q^i_7 have a range of ± 170 deg, whereas joints q^i_2, q^i_4 and q^i_6 have a range of ± 120 deg. Given the 20 deg sampling resolution, this leads to 18 samples for the former group and 13 for the latter. One can see from Figure 6 that sampling joint q^i_7 has no effect on the configuration of the spheres (even when considering a tool attached to it). Hence, the total number of possible configurations is $18^3 * 12^3$. C++ code to efficiently generate such dataset is provide in the SCA_boundary_construction package(Table 3), the user needs only to specify the X_{off} between robot bases and the DH parameters of each arm. For our *dual-arm setting* we gathered a dataset of approximately ≈ 5.4 million data-points, ≈ 2.4 million belonging to the "collided" configuration class $y = -1$ and the rest to the "non-collided" configurations $y = +1$. Due to our systematic sampling of "collided" and "boundary" robot configurations, we can generate such balanced datasets, which is desirable for any learning algorithm.

5.3 Efficient SVMs for Large Datasets

Training time of a kernel SVM has a complexity of $\approx \mathcal{O}(N_M{}^2 D)$, where N_M is the number of samples and D is the dimension of the data-points. Prediction time, on the other hand, depends on the number of support vectors N_{sv} learned through training. In practice, the N_{sv} tends to increase linearly with the amount of training data N_M (Bakir et al. 2004). More specifically, for a kernel SVM $N_{sv}/N_M \rightarrow \mathscr{B}_k$, where $\mathscr{B}_k$ is the smallest achievable classification error by the kernel k (Steinwart 2003); i.e. in a non-separable classification scenario, to achieve 5% error, at least 5% of the training points must become support vectors. This comes as a nuisance when large training sets are involved, as is the case for our application. A $N_{sv} \gg$ signifies a dense solution for representing the hyper-plane of the classifier margin $\mathbf{w} = \sum_{i=1}^{N_{sv}} \alpha_i y_i \Phi(f(q_i^{ij}))$. Naturally, the denser the solutions, the more computationally expensive they are at run-time. This makes dense SVMs infeasible for real-time robot control. In order to achieve fast adaptation for both the desired end-effector positions and self-collisions, the IK solver must run

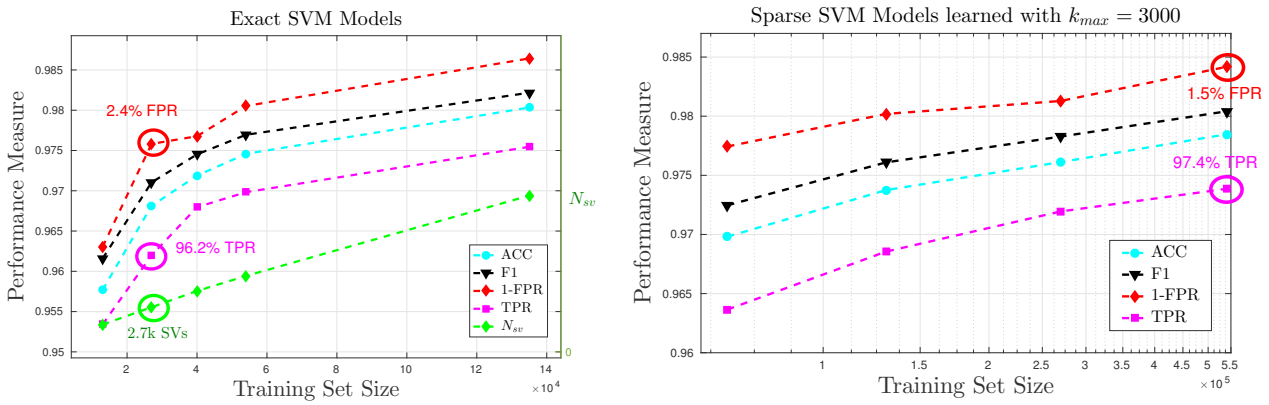


Figure 9. Performance Comparison of learning exact SVM models on *randomly sub-sampled* datasets vs sparse SVM models on larger chunks of the dataset. Each model was evaluated on the test set, which contains 2.7 million unseen sample robot configurations. We present accuracy (ACC), F-1 Score (F1), False Positive Rate (plotted as 1-FPR), True Positive Rate (TPR) and N_{sv}. (left) With the random sub-sampling method, using the 2nd model ($N_{sv} = 2,7k$), one can achieved $FPR \approx 2.4\%$ and $TPR \approx 96.19\%$ within the desired $2ms$ runtime limit. (right) With a sparse SVM model trained on $540k$ points we can achieve $FPR \approx 1.45\%$ and $TPR \approx 97.4\%$ $\mathbf{k_{max} = 3000}$.

(at most) at a rate of $2ms$. During this cycle, prior to solving (17), both (24) and (26) must be evaluated.

Given the desired control rate ($2ms$), the specific hardware used to control the robots (i.e. 3.4-GHz i7 PC with 8GB RAM) and the kinematic specifications of each robot, we can define a *computational budget* for our Self-Collision Avoidance (SCA) Boundary function. This *budget* translates to, defining a limit of the maximum allowable N_{sv} for our SVM representation of $\Gamma(f(q^{ij}))$. In Figure 8, we show a plot of different computation times[8] for the evaluation of (24) and (26) for the dual-arm setting shown in Figure 7. We omit computation time of the IK solver as this is presented in detail in Section 6 Figure 16b. These computation times evaluate the implementation of $\Gamma\left(f(q^{ij})\right)$ and $\nabla\Gamma\left(f(q^{ij})\right)$ from the C++ SVMGrad (Table 3) package, provided by the authors.

According to Figure 8, in order to comply with the $2ms$ runtime requirement, we have a *computational budget* of $N_{sv} \leq 3k$. Given the size of our dataset, it is not feasible to train SVM models that typically optimize for the dual through a variant of Sequential Minimal Optimization (SMO) (Platt 1999) or SMO-type decomposition methods (Joachims 1999; Bordes et al. 2005; Chang and Lin 2011). The fact that SVM learning algorithms tend to produce dense solutions has been recognized as one of its main weaknesses. To this end, several approaches have been proposed in order to solve the problem of finding sparser solutions to $\mathbf{w}$. These can be categorized into: i) post-processing approximations and ii) objective function or optimization strategy modification. The *former* approaches rely on approximating a sparse solution to an initially dense SVM (through the exact solution). The *latter* approaches either modify the SVM objective function by imposing sparsity constraints or propose a modified optimization algorithm for with sparsity considerations.

In this work, we choose one of the prevailing approaches which reformulates the SVM optimization problem, namely the Cutting Plane Subspace Pursuit (CPSP) method introduced by (Joachims and Yu 2009); as it directly estimates a solution to the hyper-plane with a strict bound on the number of support vectors k_{max}. A short motivation for selecting this method is discussed in Appendix H. In short, the CPSP method approximates a sparse hyper-plane by expressing it in terms of a set $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_{k_{max}}\}$ of basis vectors $\mathbf{b}_i \in \mathbb{R}^{3d_{q_i}+3d_{q_j}}$ (not necessarily training points) as follows,

$$\mathbf{w} = \sum_{i=1}^{k_{max}} \alpha_i y_i \Phi(\mathbf{b}_i). \tag{28}$$

The optimization algorithm to estimate (28) then focuses on *pursuing* such a subspace through the fixed-point iteration approach for RBF kernels (Scholkopf and Smola 2001). The learned basis vectors $\mathbf{B}$ and α_i's can be directly used in (24) and (25). We direct the interested reader to (Joachims and Yu 2009) for theoretical equivalence proofs and implementation details of this learning approach.

$\Gamma\left(f(q^{ij})\right)$ *Learning Performance:* We begin our $\Gamma\left(f(q^{ij})\right)$ learning performance analysis by presenting results from learning exact SVMs from small sub-samples of the $5.4m$ point dataset. To generate such comparison, the libSVM library (Chang and Lin 2011)[9] was used for learning the SVM models, cross-validation was performed with routines from ML_toolbox[10] to find the optimal hyper-parameters. A 50% split for training+validation/testing datasets was used to generate such evaluations. We evaluate 5 models with increasing complexity $N_{sv} = \{1.7k, 2.7k, 3.7k, 4.7k, 9.7k\}$. These were learned from using $\{0.5\%, 1\%, 1.5\%, 2\%, 5\%\}$ of the training set (i.e. 2.7 million data-points)[11]. For each model, a 10-fold Cross-validation was performed to find the optimal hyper-parameters C and σ which yield the best trade-off between N_{sv}/N_M and classification accuracy. The search space of each hyper-parameter was log-spaced in the following ranges $C = [10^{-1}, 10^4]$ and $\sigma = [0.2, 2]$[12]. In our application, we care about correctly classifying the negative class (i.e. "collided" configurations), for this we have two objectives:

- *Minimize False Positive Rate (FPR):* The $FPR = \frac{FP}{(FP+TN)}$, otherwise known as *Fall-out* error, quantifies the probability of negative samples ($y = -1$) being classified as positive ($y = +1$). This is equivalent to maximizing

the True Negative Rate ($TNR = 1 - FPR$). Classifying "collided" configurations ($y = -1$) as "non-collided" configurations ($y = +1$) yields a False Positive (FP). This error is critical as it would cause the IK solver to move the robots into an infeasible region, leading to collision and possibly permanent damage.

- *Maximize True Positive Rate (TPR):* The $TPR = \frac{TP}{(TP+FN)}$, otherwise known as *Recall* or *Sensitivity*, quantifies the probability of positive samples ($y = +1$) being classified as positive ($y = +1$). This is equivalent to minimizing the False Negative Rate ($FNR = 1 - TPR$). Classifying "collided" configurations ($y = +1$) as "non-collided" ($y = -1$) yields a False Negative (FN). This error is not as critical as the former, but it has an effect on the performance of the IK solver, as classifying "non-collided" configuration as "collided" would restrict the IK solver to move the robots into regions that are indeed feasible.

For reference, we also present accuracy $ACC = \frac{TP+TN}{(TP+FP+FN+TN)}$ and F1-Score $F1 = \frac{2TP}{(2TP+FP+FN)}$. As can be seen from Figure 9, we can achieve optimal error rates on the testing set of $FPR \approx 1.3\%$ and $TPR \approx 97.54\%$, with 5% of the training dataset, albeit surpassing the N_{SV} limit. One might argue that, with such high performance of models trained on a minuscule amount of data (relative to the complete dataset), perhaps such a large dataset is not necessary. This is related to the SCA dataset construction procedure (Section 5.2), where we set the joint sampling interval to 20deg. An analysis of the performance of SVM models trained on datasets with lower sampling resolutions is provided in Appendix G. In short, as we increase the joint sampling resolution, less "collided" configurations are seen, resulting in drastic increases in FPR.

From Figure 9, we can see that with the 2nd model (i.e. 1% of training data), we achieve error rates of: $FPR \approx 2.4\%$ and $TPR \approx 96.19\%$ withing the *computational budget*. This is quite acceptable performance, however, due to the delicacy of our application we seek to achieve the best solution possible, i.e. at least $FPR \approx 1\%$. In Figure 9, we present the results of using the CPSP SVM learning approach on different sub-sets of our training data limited to a support vector budget of $k_{max} = 3000$, specifically $\{2.5\%, 5\%, 10\%, 20\%\}$. As can be seen, for the models learned on datasets with the same size as the exact SVM solutions, the results are marginally lower. However, as the number of training-points increases the error rates improve as much as $FPR = 1.5\%$ and $TPR = 97.4\%$ for a training set of $540k$ points. By using this sparse learning method we have proven that optimal error rate can be achieved with minimal model complexity.

6 Empirical Validation

The performance of the proposed framework is implemented on two different real dual-arm platforms. On the first platform, the coordination between the arms and with the object is evaluated. The second experimental set up is designed to evaluate the performance of *dual-behavior* and the *self-collision* avoidance.

6.1 First experimental set-up

The proposed framework is implemented on a real dual-arm platform, consisting of two 7 DOF robotic arms, namely a KUKA LWR 4+ and a KUKA IIWA mounted with a 4 DOF Barrett hand and a 16 DOF Allegro hand. As the results from the first set-up were presented in our previous work, we only summarize the main points. For more information, the readers are referred to (Salehian et al. 2016a).

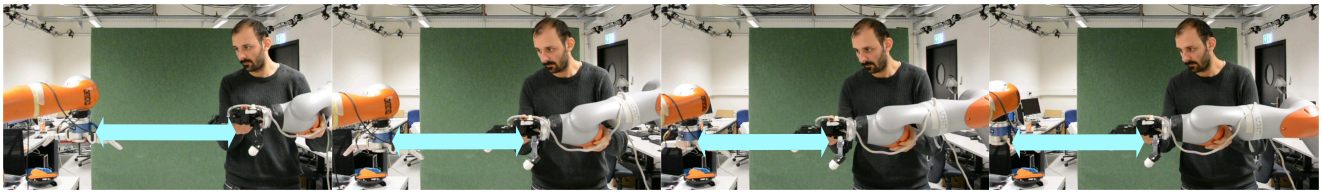
The empirical validation is divided into two parts that demonstrate the controller's ability: (i) coordinate the multi-arm systems; (ii) rapidly adapt bi-manual coordination to intercept a flying object, without using a pre-defined model of the object's dynamics. As the synchronized behavior is the only desired behavior in this section, the value of synchronization parameters are manually set to one.

6.1.1 Coordination Capabilities. The first scenario is designed to illustrate the coordination capabilities of the arms with each other through the *virtual object*. As the human operator perturbs one of the robot arms, the virtual object is perturbed as well, resulting in a stable *synchronous* motion of the other unperturbed arm (Fig. 10). Since we offer a centralized controller based on the virtual object's motion, there is no master/slave arm; thus, when any of the robots are perturbed, the others will synchronize their motions accordingly.

6.1.2 Reaching for Fast Flying Objects. The second scenario is designed to show the coordination between the robots and a fast moving object, where a rod $(150 \times 1cm)$ is thrown to the robots from $2.5m$ away, resulting in approx. $0.56s$ flying time. Due to inaccurate prediction of the object trajectory, the feasible intercept points need to be updated and redefined during the motion execution. The new feasible intercept point is chosen in the vicinity of the previous one to minimize the convergence time. As the motion of the object is fast and the predicted reaching points are not accurate, the initial values of γ in (15) are set to 0.5. This decreases the convergence duration of the robots to the real object. Snapshots of the real robot experiments are shown in Fig. 11. Visual inspection of the data and video confirmed that the robots coordinately follow the motion of the object and intercept it at the vicinity of the predicted feasible intercept point.

6.2 Second experimental set-up

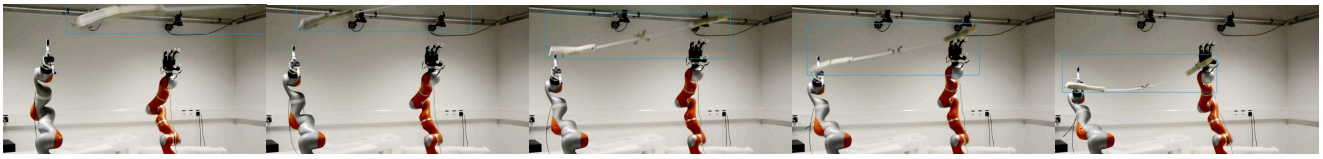
The proposed framework is implemented on a dual-arm platform, consisting of two 7 DOF KUKA IIWA robotic arms mounted with a 2 finger Robotiq gripper and a 16 DOF Allegro hand. The robots are controlled via Fast Research Interface (FRI) at joint impedance mode. The fingers are controlled with joint angle position controllers in two states: Open, Close.[13] All the hardware involved (e.g. arms and hands) are controlled by one 3.4-GHz i7 PC.[14] The position of the *feasible reaching points* of the objects are captured by an Optitrack motion capture system from Natural point at 240 Hz. As the outputs of the vision system are noisy, a Savitzky-Golay filter [15] is used to smooth the position of the object and estimate velocity and acceleration from these position measurements.



(a) (b) (c) (d)

Figure 10. Snapshots of the video illustrating coordination of the arms in free space. Both arms are manually assigned to the synchronized behavior and as the object is outside the workspace of the robots, the coordination parameter γ is close to 0. The human operator perturbs one of the arms, which leads the other arm to move in synchrony following the motion of the virtual object attached to the two end-effectors.

(a) The object flight duration is $0.39s$.



(b) The object flight duration is $0.48s$.

Figure 11. Snapshots of the arms reaching for a fast moving object. The object is specified by a blue square. The arms move in **same** direction (a) or in **opposite** directions (b) to keep the coordination between the arms and with the object. In order to not damage the robot's hands, the robot hands do not close on the object when the hands intercept the object. A corresponding video for these experiments can be found in our previous publication (Salehian et al. 2016a).

Our empirical validation is divided into two parts which demonstrate the controller's capabilities: (i) estimation of the desired behavior and accordingly adaptation of the two arms' motions; i.e. move independently or in coordination (ii) self-collision avoidance. A corresponding video is available at Multimedia Extension 1.

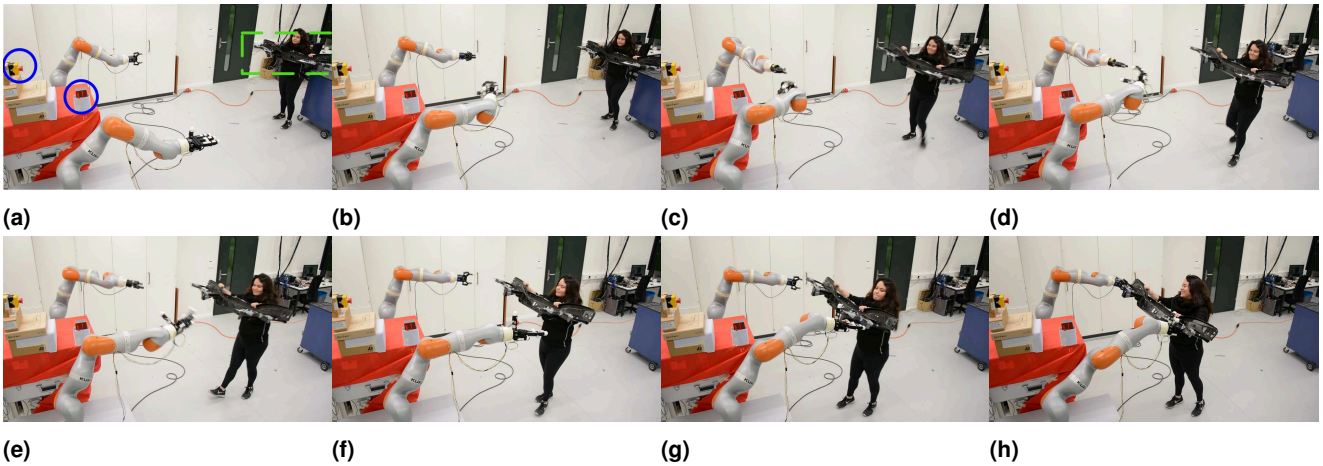
The performance of the framework is systematically assessed in three different levels to show: (i) the success rate of coordinately reaching for moving object, (ii) performance of IK solvers and (iii) sensitivity of the framework to noise.

6.2.1 Dual Behavior Capabilities. The first scenario is designed to illustrate the dual-behavior capabilities of the arms. The asynchronous behavior of each robot is to reach a fixed target (see Fig.12) or follow the hands of operator 1 who stands between the arms (see Fig.13). The synchronized behavior is to coordinately reach an object brought by operator 2. The vision inspection of the data shows that when operator 2 moves the object toward the robots, based on (8), the value of τ_{c_i} $\forall i \in \{1,2\}$ smoothly increases to one. Hence, a smooth transition from the *unsynchronized* behavior to the *synchronized* behavior is achieved; see Fig. 12 and Fig. 13. As there is no full coordination between the arms while the value of τ_{c_i} $\forall i \in \{1,2\}$ is less than one, perturbing one arm does not affect the motion of other arm, see Figure 13(a),(b),(c),(d). While the arms are allocated to the synchronized behavior, due to (15), the arms successfully track, coordinate with and intercept the object and with each other, see Figure 13(g),(h).

6.2.2 Self-Collision Avoidance. Even though, the self-collision avoidance is successfully tested in all the experiments, we designed two different scenarios to highlight the performance of the inverse kinematic solver with this constraint. As the motion of the fingers are not considered in the generated data-set, see Section 5.2, we removed the hands in these two scenarios. In the first scenario, both robots move on straight lines to reach fixed and predefined targets. The targets are picked such that the arms surely collide with each other if the collision avoidance constraint is not considered. Figure 14c shows the initial and the final configurations. As it can been seen from Figures14a the value of $\Gamma(.)$ is coordinately updated during the motion execution with respect to the robot configurations and when it is less than 2, based on (17d), the robots are moved away from each other to increase their distance. The minimum distance between the robots' segments are shown in Figure14b. In the second scenario, the end-effectors are following the hands of an operator. Visual inspection of the data and video confirmed that the robots follow the targets while any collision between them are avoided. A partial result of this experiment is presented in the accompanying video, see Multimedia Extension 1.

6.2.3 Systematic assessment

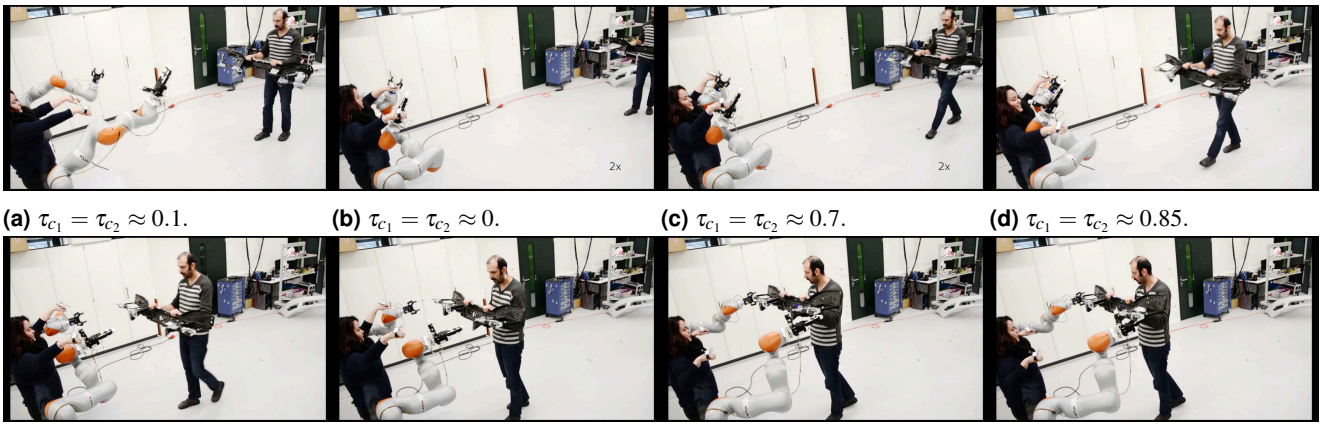
Coordination and Adaptation Assessment. To systematically assess the performance of the proposed framework. We design a handover scenario, where an operator holds an object and moves toward the arms and hand overs the object to the robots. The robots' hands are triggered to close when the distance between the arms and the object is less than $1cm$. The success rates of our experiments are measured by defining a Boolean metric; i.e. success or



(a) **(b)** **(c)** **(d)**

(e) **(f)** **(g)** **(h)**

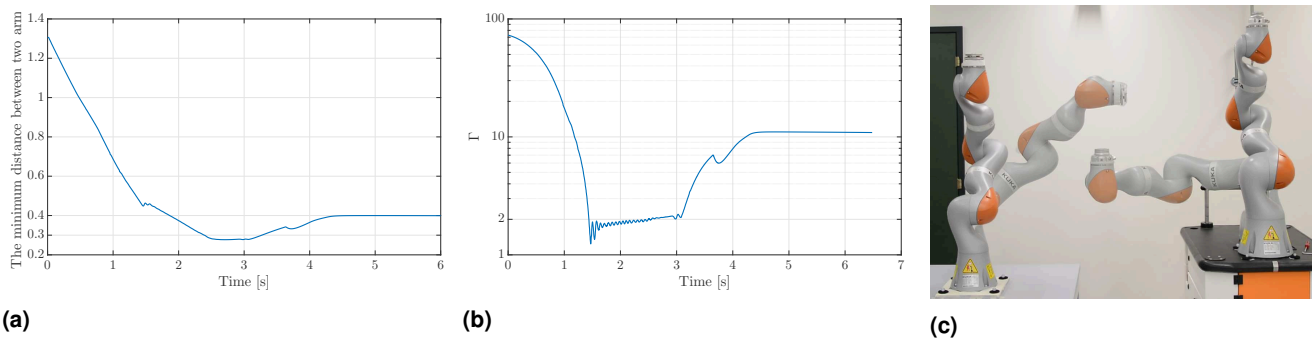
Figure 12. Snapshots of the video illustrating of dual behavior capabilities. The target of synchronous and asynchronous behaviors are highlighted in (a) by the green square and the blue circles, respectively. Initially, the robots are allocated to the asynchronous behavior. Hence, the robots moves toward the asynchronous targets in (b) and (c). In (d), $\gamma_{c_i} \approx 1 \ \ \forall i \in \{1,2\}$ as the operator moves the object toward the arms; i.e. the synchronous behavior. Consequently, the robots coordinately and simultaneously reach and intercept the object at the desired reaching points.



(a) $\tau_{c_1} = \tau_{c_2} \approx 0.1.$ **(b)** $\tau_{c_1} = \tau_{c_2} \approx 0.$ **(c)** $\tau_{c_1} = \tau_{c_2} \approx 0.7.$ **(d)** $\tau_{c_1} = \tau_{c_2} \approx 0.85.$

(e) $\tau_{c_1} = \tau_{c_2} \approx 1.$ **(f)** $\tau_{c_1} = \tau_{c_2} \approx 1.$ **(g)** $\tau_{c_1} = \tau_{c_2} \approx 1.$ **(h)** $\tau_{c_1} = \tau_{c_2} \approx 1.$

Figure 13. Snapshots of the video illustrating of dual behavior capabilities. The asynchronous behavior is to follow the hands of the operator 1 who is inside of the robot workspaces. When the operator 2 moves the object away from or toward the arms, the synchronization parameter smoothly goes to 0 and 1, respectively.

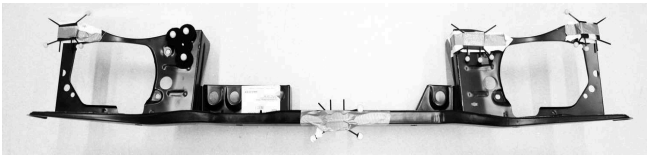


(a) **(b)** **(c)**

Figure 14. The distance between the bases of the arms are $\begin{bmatrix}0.0 & -1.3 & 0.14\end{bmatrix}^T m$. The targets are $\begin{bmatrix}0.0 & -0.6 & 0.55\end{bmatrix}^T m$ and $\begin{bmatrix}0.0 & -0.6 & 0.55\end{bmatrix}^T m$ for the arms of the left and right, respectively. In (a), the initial and the final configurations (transparent one) are depicted. (b) and (c) show the minimum distance between the arms and the value of Γ during the motion execution. As it is shown Γ never goes bellow 1 which indicates that the motion of the arms are safe.

failure. A trial is classified as a success if the robot intercepts the object at the desired point within less than $1cm$ error. We choose three car parts with different sizes and shapes to validate the algorithm, i.e. a bumper, a fender and a front panel, see fig 15. The reaching orientation for each robot is specified by the operator with respect to the orientation of the tool. The experiments were repeated 20 times for each object. The objects were moved and handed-over by the operator

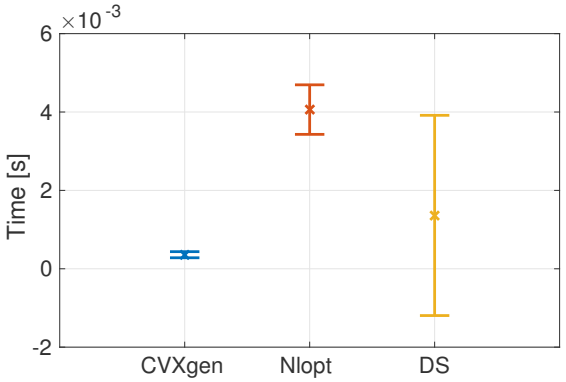
(a) The bumper.



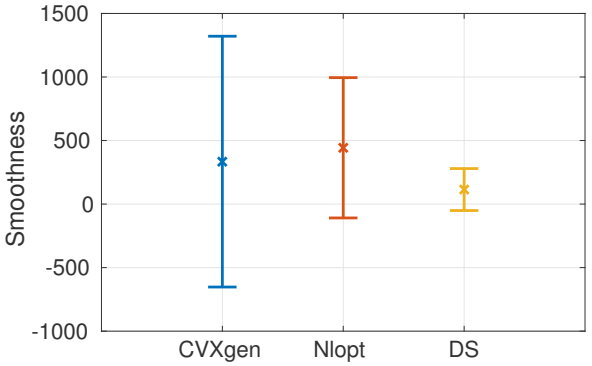
(c) The front panel.

(b) The fender.

Figure 15. The car parts which are used for the systematic assessment of the framework.



(a) Computation time comparison



(b) Smoothness of the motion comparison

Figure 16. The results of performance of the solvers for (17) in terms of computation time and the smoothness of the motion. DS stands for the dynamical system (20).

Table 1. The details of the systematically assessment experiment. All the positions are expressed with respect to the base of the KUKA IIWA 7 robot. The starting positions are randomly chosen by the operator. The robots do not move till the first intercept point is calculated, we call the position of the object at this time the first point. The first $0.4m$ of the objects' motions in x direction are used to initialize the object prediction trajectory, see Section 3.1.

	Weight	Material	The bases distance	Initial position (m)	First point (m)	Duration (s)	**Success**
Bumper	2.2	Plastic	[0.2 −1.4 0.1]	$[-3.6\pm0.2 \ -0.9\pm0.4 \ 0.6\pm0.2]$	$[-2.4\pm0.3 \ -1.0\pm0.2 \ 0.7\pm0.1]$	3.4 ± 1.6	**85%**
Front panel	2.9	Metallic	[0.2 −1.4 0.1]	$[-4.1\pm0.3 \ -0.9\pm0.2 \ 0.4\pm0.2]$	$[-2.7\pm0.4 \ -0.8\pm0.3 \ 0.6\pm0.1]$	3.2 ± 1.3	**85%**
Fender	2.4	Metallic	[0.0 −1.3 0.1]	$[-3.8\pm0.5 \ -0.7\pm0.3 \ 0.6\pm0.1]$	$[-2.7\pm0.4 \ -0.7\pm0.2 \ 0.7\pm0.1]$	2.3 ± 1.2	**90%**

from random initial positions with different orientations. The snapshots of the experiments are shown in Figure 19 and an example of the motion of the arms and the object is shown in Fig.18. The variation of the intercept points are illustrated in Fig.17. Data of the experimental results are summarized in Table 1. In this table, the first point is the position of the object when, for the first time, the *feasible* intercept point is determined. Motion duration is the duration of the objects' motions from the first point till the intercept position. The bases distances is the distance between the robots' bases.

The overall success rate of the experiment is 86.7%[16]. Failures are mostly due to inaccuracies in the measurement of the object's state. To find the position of the reaching areas and the orientation of the object, all the five markers must be visible to the cameras. The objects' tracking markers were occluded partly when the objects were covered by the robotic arms or the operator. In 5 out of 8 cases, one or two out of the five markers were not detected accurately when the object was close to the robots, hence either the robots converged to a wrong position or the synchronization parameters were changing undesirably. These two cases can be detected easily. In the first case, the hands are closed where there is no object. In the second case, the robots rapidly move back and forth. In 2 out of 8 cases, the robots started moving very late as the object predicted motion was completely wrong. In this case, the object was inside of the robots' workspaces when the robots started moving. As the motions of the objects was not extremely fast, the IK solver was able to accurately generate the joint space trajectory and only in 1 out of 8 cases it failed to track the desired end-effectors' motions. In this case, the operator suddenly changed the object's orientation when it was about to be intercepted by the robots. Hence, the reaching points on the object became kinematically infeasible for the robots to reach.

To systematically study the performance of the IK solvers and sensitivity of the framework to unmeasured object positions, two sets of simulations were designed to reach for

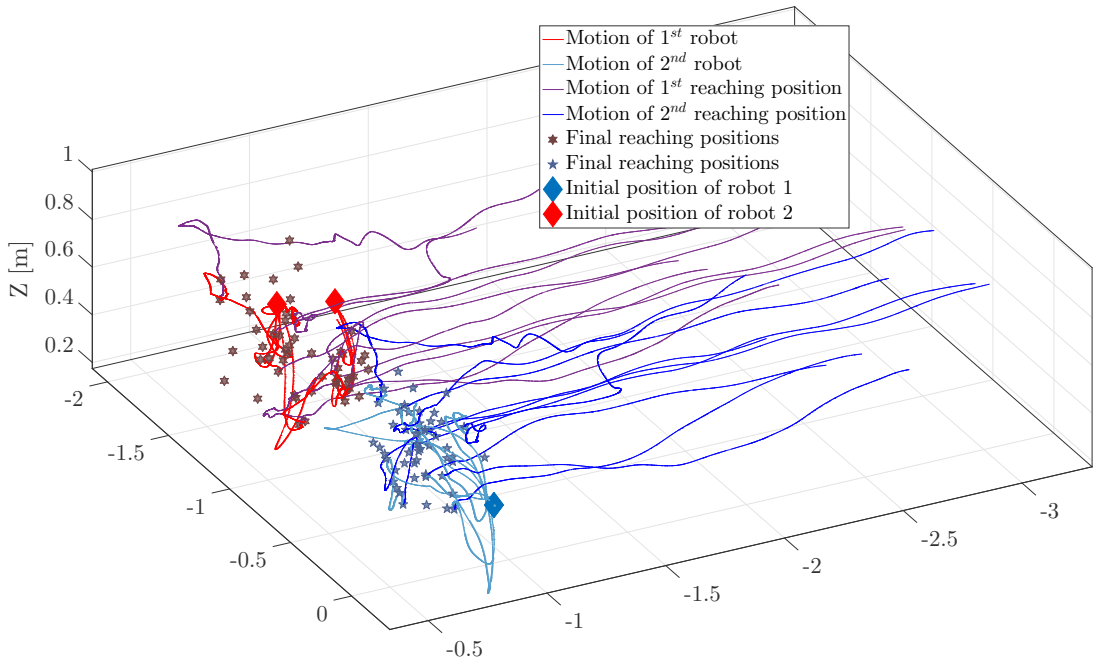


Figure 17. Spatial Variation of final intercept points. For clarity, only ten runs (i.e. trajectories) of the object and robots' motions are shown. For each run a different object trajectory and final intercept points were observed. Experimental results verified that the robots intercept the object in *synchrony*.

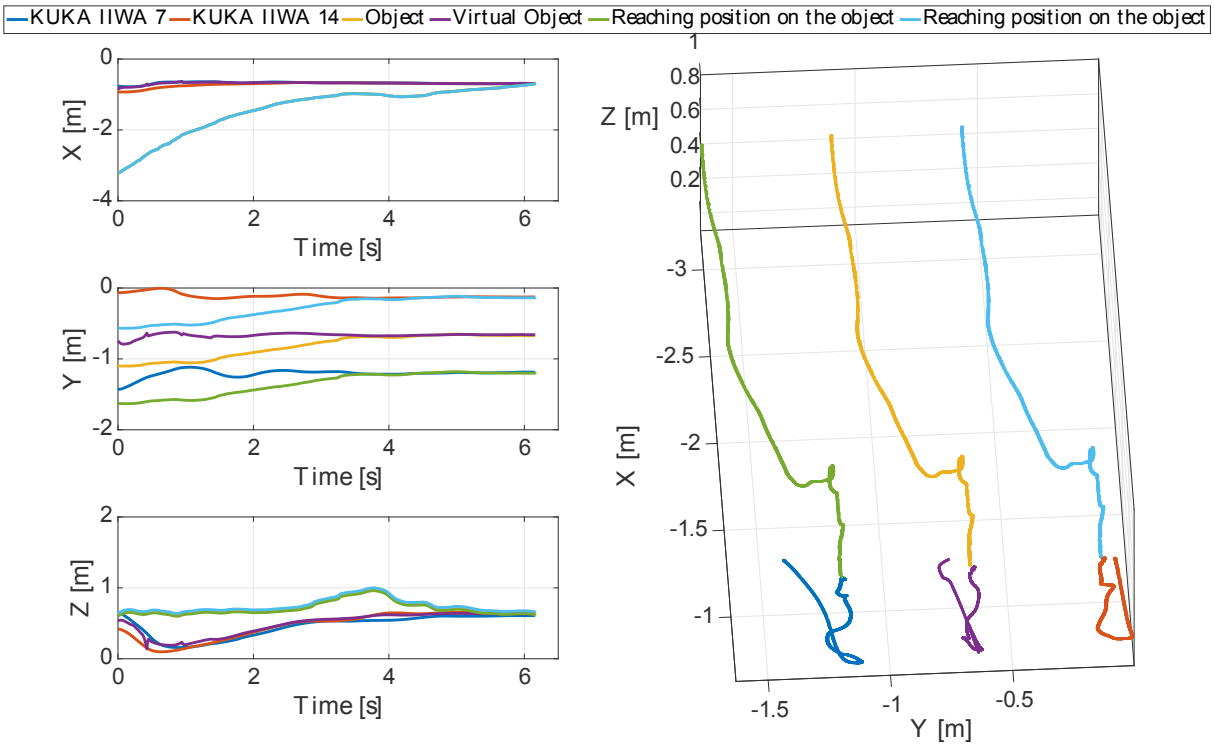


Figure 18. The position of the end-effector, the virtual object generated by (2) and (11), respectively. These trajectories are illustrated from the first point till the stop point. As expected, both arms intercept the reaching positions on the object at the same time. In order to avoid any internal forces, the robots are stopped once the fingers are closed on the object.

a moving box. The size of the box is same as the size of the bumper. In both scenarios, the object is moving toward the robots on a straight line. The simulations are conducted in the kuka-rviz environment, see Table 3.

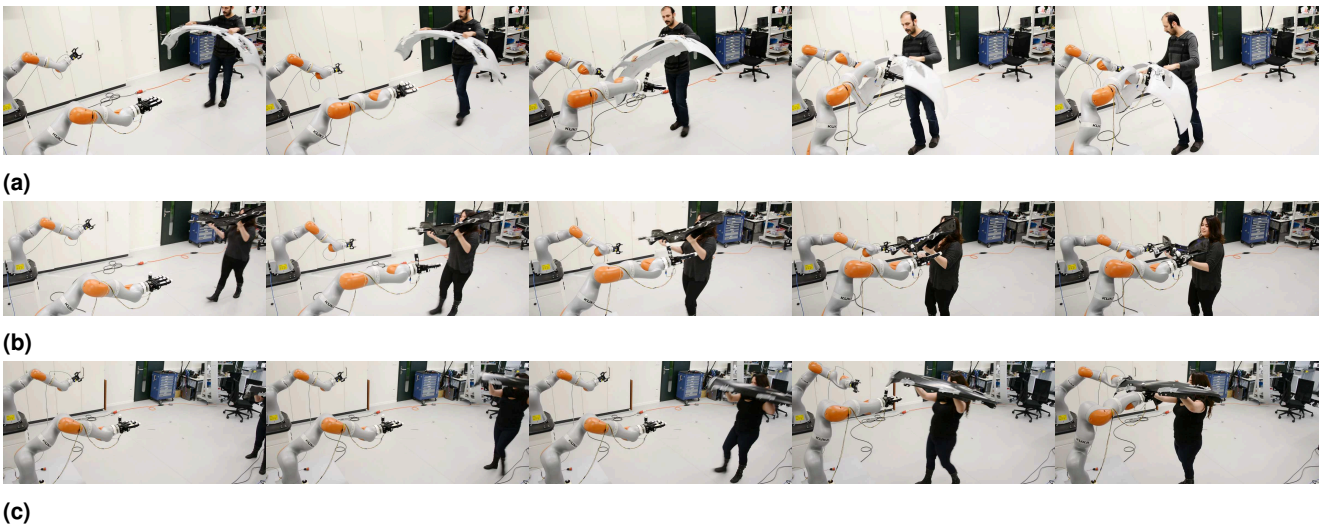


Figure 19. Snapshots from systematically assessment experiment. The objects are a bumper, a front panel and a fender in (a), (b) and (c), respectively. The robots are stopped when the hand and the gripper are closed. A corresponding video is available at Extension 2 or `https://youtu.be/S5fvr_wZ_W0`.

IK Solver Performance. In the first set of simulations, the performance of the three solvers of (17) (CVXgen, Nlopt and the dynamical system (20)) are assessed in terms of computation time and the smoothness of the generated joint motions. The initial velocity of the object is fixed but the initial position is randomly chosen within the range of $\begin{bmatrix}-3.5\pm0.05 & -0.45\pm0.05 & 0.8\pm0.05\end{bmatrix} m$. The simulation is repeated 5 times for each solver which results in more than $5*35000$ data points. The termination tolerance of the solvers is set to 10^{-4}. The computation time of each solver is illustrated in Fig.16a. As it was expected, CVXgen is the fastest solver and it takes about 0.000358s for it to solve (17) in average. The performance of the our implementation of (20) takes approximately 0.00092s to solve (17). As initialization of the dynamical system (20) plays important role in the convergence duration, the standard deviation of the computation time of (20) is much higher than other two approaches. The smoothness of the trajectory L is assessed by $\mathscr{S} = \frac{\text{std}(L)}{\text{mean}(L)}$, where a smaller value of it indicates a smoother motion. As it is shown in Fig.16b, the result of (20) is much smoother than the other methods. It was expected as (20) calculates the desired motion at the acceleration level. Hence, the output of (20) can directly be transmitted to the robots, but the outputs of either Nlpot or CVXgen need to be filtered. As the computation power was the main criterion for choosing the IK solver for us, we mostly used CVXGEN during the experiments.

Sensitivity to noise. In the second set of simulations, the robustness of the framework to noise and unmeasured object position is assessed. The simulation is repeated 165 times in total for three different object velocities; i.e. $0.25\frac{m}{s}$, $1.25\frac{m}{s}$ and $3.75\frac{m}{s}$. Results from this evaluation indicate that the interception error is directly correlated to the percentage of unmeasured object points and the velocity of the moving object (see Fig. 20). Thus, the faster the object, the more sensitive the system is to tracking inaccuracies.

7 Summary and Discussion

In this paper, we proposed a dynamical system based unified framework for generating motion of multi robotic arms to either coordinately reach a moving object or reach stably to a desired point. If provided with robotic arms that can travel fast enough, our algorithm can select the most feasible robotic arms to intercept the object in coordination and with the desired velocity aligned to the object while any collision between the arms are avoided. For selecting the most feasible robotic arms, we define a parameter (i.e the *synchronization* parameter) to assign the robots to the appropriate behavior; i.e. the *asynchronous* or *synchronous* behaviors. The *synchronization* parameter varies between zero and one based on the feasibility criteria.

The stability and convergence of the proposed dynamical system depends on ensuring conditions (13). As there are no constraints on the magnitude of the eigenvalues $(|\lambda_{A_{ij}}|)$ of A_{ij} $\forall(i,j) \in \{(1,1),(1,2),\ldots,(N_R,d_{s_{N_R}})\}$, there is no analytical proof that (11) is fast enough to converge to $\gamma\xi^O$ in time. To address this challenge, a potential direction would be to estimate the parameters of (11) with respect to the stability and the convergence rate constraints. One way to calculate the convergence rate of a dynamical system is to prove that it is exponentially stable (Khalil 2002).

To solve the quadratic programming problem, we used three different approaches. First a dynamical system based approach, Eq.(20). Second, nonlinear programming solver (Johnson 2016). The third approach was CVXGEN, introduced in (Mattingley and Boyd 2012). Each of these approaches has its own advantages and disadvantages. Using the first approach is advantageous in the way that the passivity of the dynamical system can be proven. Hence, the unified framework stays passive and stable as long as the robots are passive. In addition, the first approach result in smoother joint motions. The main advantage of the second approach is its interface. Nlopt is very user friendly and it is possible to test several solvers, but it is computationally costly. The main advantage of the third approach is the computational cost. As it has been shown in (Mattingley

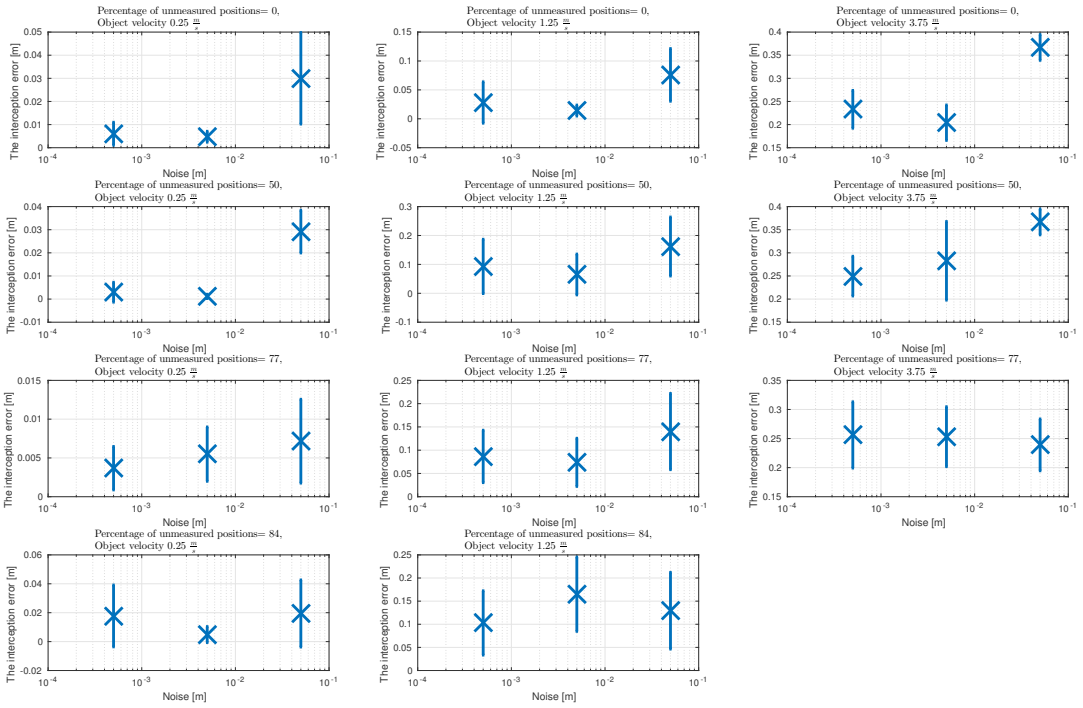


Figure 20. The interception error is the average of the minimum distance between the box and the end-effectors. The initial positions of the box are randomly chosen within the range of $\begin{bmatrix}-3.5\pm0.1 & -0.45.0\pm0.1 & 0.8\pm0.1\end{bmatrix}^T m$. The distance between the arms and the size of the box are same as the bumper scenario. The simulations are repeated for each combination of three object's speeds, three noise powers and four percentages of unmeasured positions. We only consider trials when the box passes through the robots workspaces. The measurement noise is simulated with pseudo-random values within the range of ±0.05, ±0.005 and ±0.0005m. The results of the worse case are not illustrated as the robots were not able to follow the object; i.e. the worse case is when the percentage of unmeasured position is 85% and the box's velocity is $3.75\frac{m}{s}$.

and Boyd 2012), CVXGEN is computationally very efficient. The main shortcoming of the third approach is the stability of the closed loop system which can not be proven; however we have not seen any unstable behaviors during the real world evaluations or the simulations.

Throughout the proofs, we assume that the intercept point is a fixed attractor. However, due to the imperfect prediction of the object trajectory, the feasible intercept postures need to be iteratively updated. Nevertheless, this does not affect the convergence of the system for two main reasons. First, when $\gamma < 1$ the new feasible intercept point is chosen in the vicinity of the previous ones; i.e. the convergence rate is much faster than the rate of update. Second, when the object is reachable, $\gamma = 1$, the virtual object converges to the real object and the position of the intercept point does not affect the convergence. If we assume that the object trajectory is predictable, one can simply generalize a single-robot arm feasible posture extraction (Kim et al. 2014) for multiple arms to extract the intercept posture. However, the proposed algorithm is not restricted to this assumption. In this work, the feasible intercept point is calculated to make sure that, firstly, the object is passing through the robots' reachable space and, secondly, to move the robots in advance to the vicinity of this point to avoid high accelerated motions.

In this paper, we control the motion of the arm from initial condition (palm open, robots far from the object) to the point when the arms reach the object and the fingers are about to close on the object. Hence, there are no interaction forces (which would arise once in contact with the object). Once the fingers close on the object, the robots-object system become a closed kinematic chain. In this case, devising an appropriate force controller is necessary to coordinate the robots. Future work in multi-arm manipulation will be directed to address the challenges of devising an appropriate force controller and the transition between the position and force controller.

One of the main advantages of the proposed framework is the computational cost. The implementation shows that the overall computation is rapid, thus enabling us to not only compute the feasible intercept point, the reaching motion and solve centralized IK problem, but also to control a 33 $(7+7+16+1)$ DOF system with one 3.4-GHz i7 PC.

Finally, we are currently working on improving the performance of (15), by learning its parameters via a convex optimization problem with respect to the workspace constraint of the robots. With this approach, we could ensure that the performance of the dynamical system is optimal and the generated motion is not infeasible for the robots to follow.

Acknowledgements

Research leading to this work was supported by the EU project *Cogimon* $H2020-ICT-23-2014$. The implementation and evaluation presented in this work has been partly supported by KUKA in the scope of the KUKA Innovation Award 2017. The authors kindly thank Leonardo Urbano for his help during the experiments.

Notes

1. The motion generator is fully observable.
2. We set $0 \ll \mathbf{k}$ as we are mainly interested to have only two behaviors.
3. We assume that the dynamical system (11) is fast enough to converge to an acceptable neighborhood around the desired trajectory $\gamma\begin{bmatrix}\xi^O(t) & \dot{\xi}^O(t)\end{bmatrix}^T$ before T^*; i.e. $\|\xi^V(T^*) - \gamma\xi^O(T^*)\| \leq \varepsilon$
4. As an example, $N_R = 2$, $\mathbf{q} = q^{12}$. Based on our experience, due to hardware limitations, it is not possible to construct a data set of collision boundaries for more than two 7-DOF KUKA arms at once; e.g. for 3 arms, the size of the data set approximately is $(3*7*3)\times 1000^3$ while it is $(3*7*2)\times 1000^2$ for 2 arms.
5. The formulation of the Lyapunov stability proof is inspired from (Xia and Wang 2000; Zhang et al. 2004; Zhang 2005).
6. This fact has been implicitly proven in (Fang et al. 2015). The authors learned SVMs on joint-angle data q^{ij} for limb-pairs of a humanoid robot, to classify safe/dangerous configuration regions. The reported misclassification rate was between $15-35\%$, they thus relied on a Quadratic Lagrangian Interpolation (QLI) function to *improve* the classification error on-line. Since "collided" and "non-collided" configurations can differ by < 5deg rotation of 1 joint angle, the underlying binary classification problem is that of an extremely overlapped dataset. For an SVM to yield high-accuracy the model must be highly complex, i.e. large number of support vectors. (Fang et al. 2015) reported a training set size of $\approx 6k$ points. We believe they had such a small training set in order to keep the complexity of the SVM low for runtime, at the cost of low prediction accuracy.
7. For sake of simplicity, the superscripts (ij) of α^{ij}, N_{sv}^{ij}, y^{ij} and σ^{ij} are dropped in the paper.
8. This runtime includes the construction of the feature vector $f(q^{ij})$ as well as the construction of $J(q^{ij})$ which must be multiplied by (25), this involves vector and matrix constructions as well as matrix multiplication. The `http://eigen.tuxfamily.org/index.php?title=Main_Page` is used for such operation, which has underlying dynamic allocation strategies, this induces the std. seen in Figure 8.
9. libSVM: `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`
10. ML_toolbox: `https://github.com/epfl-lasa/ML_toolbox`
11. For practical reasons, we stopped evaluating models above 10% of the training data. As 10% of the training data entails learning an SVM from 270k points, not only did this take days to solve the dual QP problem, but the N_{sv} is already infeasible for robot control. One could use *faster* solvers, such as the one implemented in `http://svmlight.joachims.org/` which offers computational improvements on libSVM, however, this method can only handle up to several 100k training points. The authors of LASVM propose an active learning approach for training set selection which is claimed to be 3 times faster than libSVM and has been tested on datasets of several million points with models of marginally lower N_{sv} than libSVM(Bordes et al. 2005). On the other hand, ensemble methods have been proposed to learn kernel SVMs from *very* large datasets. `http://homes.esat.kuleuven.be/~claesenm/ensemblesvm/` parallelizes the learning of ensemble base models and applies basic aggregation schemes to generate a final SVM. This dramatically reduces training time, yet, at the cost of obtaining more support vectors than libSVM (due to model aggregation) (Claesen et al. 2014). These approaches, although capable of handling *large* datasets, focus mainly on improving training time (i.e. efficient learning), not necessarily on generating sparse solutions to the QP problem.
12. The range of σ was determined by the values of tails generated by a fitted Gaussian distribution on the pair-wise Euclidean norm of all data-points.
13. It is important to clarify that each DOF of the hand is separately controlled. For each state (open/close) a target joint configuration is defined. Once the states are triggered, we use a PD-joint position controller to guide the fingers to the desired configurations.
14. Due to implementation constraints, one of the arms are connect to another PC. Apart from this connection, no computation is done on the other PC.
15. SG_differentiation `https://github.com/epfl-lasa/sg_differentiation.git`
16. The cases which the operator moved outside of the robots' worksapces are excluded.

References

(2014) rviz 3d visualization tool for ros. URL `http://wiki.ros.org/rviz`.

Aghili F (2013) Adaptive control of manipulators forming closed kinematic chain with inaccurate kinematic model. *IEEE/ASME Transactions on Mechatronics,* 18(5): 1544–1554.

Bai H and Wen JT (2010) Cooperative load transport: A formation-control perspective. *IEEE Transactions on Robotics* 26(4): 742–750.

Bakir GH, Bottou L and Weston J (2004) Breaking svm complexity with cross-training. In: *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04. Cambridge, MA, USA: MIT Press, pp. 81–88.

Bishop MC (2007) *Pattern Recognition and Machine Learning*. Springer. ISBN 978-0387310732.

Bordes A, Ertekin S, Weston J and Bottou L (2005) Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research* 6: 1579–1619.

Caccavale F, Chiacchio P, Marino A and Villani L (2008) Six-dof impedance control of dual-arm cooperative manipulators. *IEEE/ASME Transactions on Mechatronics* 13(5): 576–586.

Caccavale F and Uchiyama M (2016) *Springer Handbook of Robotics, Chapter 39, Cooperative Manipulation*. Cham: Springer International Publishing, pp. 989–1006.

Calvin S and Jirsa VK (2011) Perspectives on the dynamic nature of coupling in human coordination. In: Huys R and Jirsa VK (eds.) *Nonlinear Dynamics in Human Behavior*, *Studies in Computational Intelligence*, volume 328. pp. 91–114.

Chang CC and Lin CJ (2011) Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2(3): 27:1–27:27. DOI:10.1145/1961189.1961199.

Chiacchio P and Chiaverini S (eds.) (1998) *Complex Robotic Systems,Chapter 3 Kinematic control of dual-arm systems.*

Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-40904-5, pp. 79–97. DOI:10.1007/BFb0035185.

Chrètien B, Escande A and Kheddar A (2016) Gpu robot motion planning using semi-infinite nonlinear programming. *IEEE Transactions on Parallel and Distributed Systems* 27(10): 2926–2939.

Chung SJ and Slotine JJE (2009) Cooperative robot control and concurrent synchronization of lagrangian systems. *IEEE Transactions on Robotics* 25(3): 686–700.

Claesen M, Smet FD, Suykens JA and Moor BD (2014) Ensemblesvm: A library for ensemble learning using support vector machines. *Journal of Machine Learning Research* 15: 141–145.

Coats RO and Wann JP (2012) Reaching a better understanding of the control of bimanual movements in older adults. *PLoS ONE* 7(10): e47222.

Cotter A and Srebro N (2013) Learning optimally sparse support vector machines. In: *In ICML*.

Escande A, Miossec S, Benallegue M and Kheddar A (2014) A strictly convex hull for computing proximity distances with continuous gradients. *IEEE Transactions on Robotics* 30(3): 666–678.

Escande A, Miossec S and Kheddar A (2007) Continuous gradient proximity distance for humanoids free-collision optimized-postures. In: *2007 7th IEEE-RAS International Conference on Humanoid Robots*. pp. 188–195.

Fang C, Rocchi A, Hoffman EM, Tsagarakis NG and Caldwell DG (2015) Efficient self-collision avoidance based on focus of interest for humanoid robots. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. pp. 1060–1066.

Franz E, Zelaznik HN and McCabe G (1991) Spatial topological constraints in a bimanual task. *Acta Psychologica* 77(2): 137–151.

Gams A, Ude A and Morimoto J (2015) Accelerating synchronization of movement primitives: Dual-arm discrete-periodic motion of a humanoid robot. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. pp. 2754–2760. DOI:10.1109/IROS.2015.7353755.

Ge SS and Cui YJ (2000) New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation* 16(5): 615–620. DOI:10.1109/70.880813.

Gharbi M, Corts J and Simon T (2009) Roadmap composition for multi-arm systems path planning. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 2471–2476.

Haken H, Kelso J and Bunz H (1985) A theoretical model of phase transitions in human hand movements. *Biological Cybernetics* 51: 347–356.

Holladay R and Srinivasa S (2016) Distance metrics and algorithms for task space path optimization. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Joachims T (1999) Making large-scale SVM learning practical. In: Schölkopf B, Burges C and Smola A (eds.) *Advances in Kernel Methods - Support Vector Learning*, chapter 11. Cambridge, MA: MIT Press, pp. 169–184.

Joachims T and Yu CNJ (2009) Sparse kernel svms via cutting-plane training. *Mach. Learn.* 76(2-3): 179–193.

Johnson SG (2016) The NLopt nonlinear-optimization package. URL `http://ab-initio.mit.edu/nlopt`.

Kalakrishnan M, Chitta S, Theodorou E, Pastor P and Schaal S (2011) Stomp: Stochastic trajectory optimization for motion planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, May 9-13.

Kanehiro F, Yoshida E and Yokoi K (2012) Efficient reaching motion planning and execution for exploration by humanoid robots. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 1911–1916.

Kelso JA (1984) Phase transitions and critical behavior in human bimanual coordination. *Am J Physiol Regul Integr Comp Physiol* 246(6): R1000–1004.

Kelso JAS, Southard DL and Goodman D (1979) On the coordination of two-handed movements. *Journal of Experimental Psychology: Human Perception and Performance* 5(2): 229.

Khalil H (2002) *Nonlinear Systems*. Prentice Hall. ISBN 0-13-067389-7.

Kim S, Shukla A and Billard A (2014) Catching objects in flight. *IEEE Transactions on Robotics* 30(5): 1049–1065. DOI:10.1109/TRO.2014.2316022.

LaValle SM (2006) *Planning Algorithms*. New York, NY, USA: Cambridge University Press. ISBN 0521862051.

Likar N, Nemec B and lajpah L (2013) Virtual mechanism approach for dual-arm manipulation. *Robotica* FirstView: 1–16.

Liu YH and Arimoto S (1998) Decentralized adaptive and nonadaptive position/force controllers for redundant manipulators in cooperations. *International Journal of Robotics Research* 17(3): 232–247.

Luh JYS and Zheng YF (1987) Constrained relations between two coordinated industrial robots for motion control. *International Journal of Robotics Research* 6(3): 60–70.

Mattingley J and Boyd S (2012) Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering* 13(1): 1–27.

Murray S, Floyd-Jones W, Qi Y, Sorin D and Konidaris G (2016) Robot motion planning on a chip. In: *Proceedings of Robotics: Science and Systems*. AnnArbor, Michigan.

Orthey A and Stasse O (2013) Towards reactive whole-body motion planning in cluttered environments by precomputing feasible motion spaces. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. pp. 274–279.

Petrič T, Gams A, Likar N and Žlajpah L (2015) *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches, Part II, Chapter Obstacle Avoidance with Industrial Robots*. Cham: Springer International Publishing. ISBN 978-3-319-14705-5, pp. 113–145. DOI: 10.1007/978-3-319-14705-5_5.

Platt JC (1999) Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization. Cambridge, MA, USA: MIT Press. ISBN 0-262-19416-3, pp. 185–208.

Ratliff N, Toussaint M and Schaal S (2015) Understanding the geometry of workspace obstacles in motion optimization. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 4202–4209.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In: *2009 IEEE International Conference on Robotics and*

Automation. pp. 489–494.

Rojas R (1996) *Neural Networks: A Systematic Introduction*. New York, NY, USA: Springer-Verlag New York, Inc. ISBN 3-540-60505-3.

Salehian SSM, Figueroa N and Billard A (2016a) Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty. In: *Proceedings of Robotics: Science and Systems*.

Salehian SSM, Figueroa N and Billard A (2017) Dynamical System-based Motion Planning for Multi-Arm Systems: Reaching for moving objects. In: *International Joint Conference on Artificial Intelligence*.

Salehian SSM, Khoramshahi M and Billard A (2016b) A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics* 32(2): 462–471. DOI:10.1109/TRO.2016.2536749.

Santis AD, Albu-Schaffer A, Ott C, Siciliano B and Hirzinger G (2007) The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In: *2007 IEEE/ASME international conference on advanced intelligent mechatronics*. pp. 1–6.

Scholkopf B and Smola AJ (2001) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press. ISBN 0262194759.

Smith C, Karayiannidis Y, Nalpantidis L, Gratal X, Qi P, Dimarogonas DV and Kragic D (2012) Dual arm manipulationa survey. *Robotics and Autonomous Systems* 60(10): 1340 – 1353.

Steinwart I (2003) Sparseness of support vector machines. *J. Mach. Learn. Res.* 4: 1071–1105.

Stilman M (2007) Task constrained motion planning in robot joint space. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 3074–3081.

Suda R, Kosuge K and Kakuya H (2003) Object-impedance-based cooperative handling of object by mobile robot helper and human using visual and force information. In: *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, volume 1. pp. 592–597 vol.1. DOI:10.1109/AIM.2003.1225161.

Sugiura H, Gienger M, Janssen H and Goerick C (2007) Real-time collision avoidance with whole body motion control for humanoid robots. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 2053–2058.

Swinnen SP (2002) Intermanual coordination: from behavioural principles to neural-network interactions. *Nature Reviews Neuroscience* 3(5): 348–359.

Vahrenkamp N, Asfour T and Dillmann R (2012) Simultaneous grasp and motion planning: Humanoid robot armar-iii. *IEEE Robotics Automation Magazine* 19(2): 43–57.

Vahrenkamp N, Do M, Asfour T and Dillmann R (2010) Integrated grasp and motion planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 2883–2888.

Vernon D, von Hofsten C and Fadiga L (2011) *A Roadmap for Cognitive Development in Humanoid Robots*, *Cognitive Systems Monographs*, volume 11. Springer Berlin Heidelberg.

Wang M, Luo M, Li T and Ceccarelli M (2015) A unified dynamic control method for a redundant dual arm robot. *Journal of Bionic Engineering* 12(3): 361 – 371.

Williams D and Khatib O (1993) The virtual linkage: a model for internal forces in multi-grasp manipulation. In: *IEEE International Conference on Robotics and Automation*. pp. 1025–1030 vol.1.

Wimböck T and Ott C (2012) Dual-arm manipulation. In: *Towards Service Robots for Everyday Environments*, *Springer Tracks in Advanced Robotics*, volume 76. Springer Berlin Heidelberg, pp. 353–366.

Wimböck T, Ott C, Albu-Schäffer A and Hirzinger G (2012) Comparison of object-level grasp controllers for dynamic dexterous manipulation. *The International Journal of Robotics Research* 31(1): 3–23.

Wimbock T, Ott C and Hirzinger G (2008) Analysis and experimental evaluation of the intrinsically passive controller (ipc) for multifingered hands. In: *IEEE International Conference on Robotics and Automation*. pp. 278–284.

Xia Y and Wang J (2000) A recurrent neural network for solving linear projection equations. *Neural Networks* 13(3): 337–350. Cited By 95.

Zhang Y (2005) On the lvi-based primal-dual neural network for solving online linear and quadratic programming problems. In: *Proceedings of the 2005, American Control Conference, 2005.* pp. 1351–1356 vol. 2.

Zhang Y, Ge S and Lee T (2004) A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34(5): 2126–2132.

Zhu WH (2005) On adaptive synchronization control of coordinated multirobots with flexible/rigid constraints. *IEEE Transactions on Robotics* 21(3): 520–525.

Zucker M, Ratliff N, Dragan A, Pivtoraiko M, Klingensmith M, Dellin C, Bagnell JAD and Srinivasa S (2013) Chomp: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research* .

A Index to Multimedia Extensions

Extension	Type	Description
1	Video	Robot experiments: Dual behavior capabilities and SCA
2	Video	Robot experiments: Systematic assessment

B Proof of Theorem 1, Part A

We propose a Lyapunov function

$$V = \frac{1}{2}(x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + (\tau_{c_i}(t) - 1)x_i^d)^T P_i^R \\ (x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + (\tau_{c_i}(t) - 1)x_i^d) \quad (29)$$

V is positive definite, radially unbounded, continuous, and continuously differentiable. The derivative of V with respect to time is

$$\begin{aligned}\dot{V} = \frac{dV}{dt} =& \frac{1}{2}\Big(\big((\dot{x}_i^R(t) - \dot{\tau}_{c_i}(t)x_i^V(t) - \tau_{c_i}(t)\dot{x}_i^V(t) + \\ & \dot{\tau}_{c_i}(t)x_i^d)^T P_i^R (x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + \\ & (\tau_{c_i}(t)-1)x_i^d)\big) + \big(x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + \\ & (\tau_{c_i}(t)-1)x_i^d)^T P_i^R (\dot{x}_i^R(t) - \dot{\tau}_{c_i}(t)x_i^V(t) \\ & - \tau_{c_i}(t)\dot{x}_i^V(t) + \dot{\tau}_{c_i}(t)x_i^d)\big)\Big)\end{aligned} \tag{30}$$

By substituting (2) into (30), we have:

$$\begin{aligned}\dot{V} =& \frac{1}{2}\Big(\big(\mathbf{A}_i(\theta_i)(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))\big)^T P_i^R \\ & (x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + (\tau_{c_i}(t)-1)x_i^d)\big) + \\ & \big(x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + (\tau_{c_i}(t)-1)x_i^d)^T P_i^R \\ & (\mathbf{A}_i(\theta_i)(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))\big)\Big) \\ =& \frac{1}{2}\Big((x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))^T \sum_{k=1}^{d_{s_i}} \theta_{ik} A_{ik}^T P_i^R \\ & (x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))\big) + \\ & \big(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))^T P_i^R \\ & (\sum_{k=1}^{d_{s_i}} \theta_{ik} A_{ik}^T (x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d)))\Big) \\ =& (x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))^T \\ & \sum_{k=1}^{d_{s_i}} \underbrace{\theta_{ik}}_{>0} \Big(\underbrace{A_{ik}^T P_i^R + P_i^R A_{ik}}_{\prec -Q_i^R} \Big) \\ & (x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d)\big) \\ & \leq 0\end{aligned} \tag{31}$$

Therefore, dynamical system (2) is globally stable; i.e. $x_i^R(t) - x_i^d - \tau_{c_i}(x_i^V - x_i^d))x_i^d$ and its time derivative is bounded. Since $\ddot{V}$ is finite, Barbalat's lemma (Khalil 2002) indicates that the attractor is globally asymptotically stable; i.e:

$$\lim_{t\to\infty} \| x_i^R(t) - \tau_{c_i}(t)x_i^V(t) + (\tau_{c_i}(t)-1)x_i^d \| = 0 \tag{32}$$

■, c.q.f.d.

C Proof of Theorem 1, Part B

Consider the following storage function:

$$V_i = \frac{1}{2}(x_i^R)^T P_i^R (x_i^R) \quad \forall i \in \{1,\dots,N_R\} \tag{33}$$

Clearly (33) is positive definite, radially unbounded, continuous and continuously differentiable. To simplify the notations, in this section, we consider $\mathbf{Y}_i = P_i^R x_i^R$ and $\mathbf{U}_i = \dot{\tau}_{c_i}(x_i^V - x_i^d) + \tau_{c_i}\dot{x}_i^V - \mathbf{A}_i(.)(x_i^d + \tau_{c_i}(x_i^V - x_i^d))$ as the output and the input of (2). To prove the passivity of (33), we need to show that

$$\frac{dV}{dt} + \psi(.) \leq \mathbf{U}_i{}^T \mathbf{Y}_i \quad \exists \psi(.),\ 0 \leq \psi(.) \tag{34}$$

The derivative of V with respect to time is as follow:

$$\begin{aligned}\dot{V} =& \frac{1}{2}(\dot{x}_i^R)^T P_i^R (x_i^R) + \frac{1}{2}(x_i^R)^T P_i^R (\dot{x}_i^R) \\ =& \frac{1}{2}\Big(\dot{\tau}_{c_i}(x_i^V - x_i^d) + \tau_{c_i}\dot{x}_i^V + \\ & \sum_{k=1}^{d_{s_i}} \theta_{ik} A_{ik}(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))\Big)^T P_i^R (x_i^R) + \\ & \frac{1}{2}(x_i^R)^T P_i^R \Big(\dot{\tau}_{c_i}(x_i^V - x_i^d) + \tau_{c_i}\dot{x}_i^V + \\ & \sum_{k=1}^{d_{s_i}} \theta_{ik} A_{ik}(x_i^R - x_i^d - \tau_{c_i}(x_i^V - x_i^d))\Big) \\ =& (x_i^R)^T \sum_{k=1}^{d_{s_i}} \underbrace{\theta_{ik}}_{0\geq} \underbrace{(A_{ik}^T P_i^R + P_i^R A_{ik})}_{\prec -Q_i^R} x_i^R + \mathbf{U}_i^T \mathbf{Y}_i\end{aligned} \tag{35}$$

Hence, $\psi(.) = -(x_i^R)^T \sum_{k=1}^{d_{s_i}} \theta_{ik}(A_{ik}^T P_i^R + P_i^R A_{ik}) x_i^R$. Hence (34) is satisfied. Furthermore, as the memoryless system $x_i^R = (P_i^R)^{-1}\mathbf{Y}_i$ is passive, the dynamical system given by (2) is passive when $\dot{\tau}_{c_i}(x_i^V - x_i^d) + \tau_{c_i}\dot{x}_i^V - \mathbf{A}_i(\theta_i(x_i^R))(x_i^d \tau_{c_i}(x_i^V - x_i^d))$and $x_i^R(t)$ are the input the output, respectively. ■, c.q.f.d.

D Proof of Theorem 2, Part A

As $\dot{x}_i^V = \dot{x}^V, \forall i \in \{1,\dots,N_R\}$, (11) can be written as:

$$\dot{x}^V(t) = \gamma\dot{x}^O + \dot{\gamma}x^O + A^V(x^V - \gamma x^O) \tag{36}$$

We propose a Lyapunov function as follows:

$$V = \frac{1}{2}(x^V(t) - \gamma x^O)^T P^V (x^V(t) - \gamma x^O) \tag{37}$$

V is positive definite, radially unbounded, continuous and continuously differentiable. Substituting, (36) into the derivative of V with respect to time results in:

$$\begin{aligned}\dot{V} =& \frac{dV}{dt} = \frac{1}{2}\Big((x^V(t) - \gamma x^O)^T P^V A^V (x^V(t) - \gamma x^O) \\ & + (x^V(t) - \gamma x^O)^T A^{V^T} P^V (x^V(t) - \gamma x^O)\Big) \\ =& (x^V(t) - \gamma x^O)^T \underbrace{(P^V A^V + A^{V^T} P^V)}_{-Q^V} (x^V(t) - \gamma x^O) \\ \leq& 0\end{aligned} \tag{38}$$

Therefore, dynamical system (36) and (11) are globally stable; i.e. x^V and $\dot{x}^V$ are bounded as γx^O, $\dot{\gamma}x^O$ and $\gamma\dot{x}^O$ are bounded. Since $\ddot{V}$ is finite, Barbalat's lemma (Khalil 2002) indicates that the attractor is globally asymptotically stable;

i.e:

$$\lim_{t\to\infty} \|\xi^V(t) - \gamma(t)\xi^O(t)\| = 0$$
$$\lim_{t\to\infty} \|\dot{\xi}^V(t) - (\gamma(t)\dot{\xi}^O(t) + \dot{\gamma}(t)\xi^O(t))\| = 0 \quad (39)$$

■, c.q.f.d.

E Proof of Theorem 2, Part B

Consider the following storage function:

$$V = \frac{1}{2}(x^V)^T P^V (x^V) \quad (40)$$

Clearly (40) is positive definite, radially unbounded, continuous and continuously differentiable. To simplify the notations, in this section, we consider $\mathbf{Y} = P^V x^V$ and $\mathbf{U} = \gamma\dot{x}^O + \dot{\gamma}x^O - A^V\gamma x^O$ as the output and the input of (11). To prove the passivity of (40), we need to show that

$$\frac{dV}{dt} + \psi(.) \leq \mathbf{U}^T\mathbf{Y} \quad \exists\psi(.),\ 0 \leq \psi(.) \quad (41)$$

The derivative of V with respect to time is as follow:

$$\begin{aligned}
\dot{V} &= \frac{1}{2}(\dot{x}^V)^T P^V (x^V) + \frac{1}{2}(x^V)^T P^V (\dot{x}^V) \\
&= \frac{1}{2}\left(\gamma\dot{x}^O + \dot{\gamma}x^O + A^V(x^V - \gamma x^O)\right)^T P^V x^V + \\
&\frac{1}{2}(x^V)^T P^V \left(\gamma\dot{x}^O + \dot{\gamma}x^O + A^V(x^V - \gamma x^O)\right) \\
&= (x^V)^T \underbrace{(P^V A^V + (A^V)^T P^V)}_{\prec -Q^V} x^V + \mathbf{U}^T\mathbf{Y}
\end{aligned} \quad (42)$$

Hence, $\psi(.) = -(x^V)^T(P^V A^V + (A^V)^T P^V)x^V$. Furthermore, as the memoryless system $x^V = (P^V)^{-1}\mathbf{Y}$ is passive, the dynamical system given by (36) and consequently (11) are passive when $\gamma\dot{x}^O + \dot{\gamma}x^O - A^V\gamma x^O$and x^V are the input the output, respectively. ■, c.q.f.d.

F Proof of Theorem 3

Consider the following storage function.

$$V = \frac{1}{2}u^T u \quad (43)$$

(43) is positive definite, radially unbounded, continuous and continuously differentiable. To simplify the notations, in this section, we consider $\mathbf{Y} = u$ and $\mathbf{U} = (I + M^T)P_\Omega(u - (Mu + b))$ as the output and the input of (20). To prove the passivity of (20), we need to show that

$$\frac{dV}{dt} + \psi(u) \leq Y^T\mathbf{U} \quad \exists\psi(u),\ 0 \leq \psi(u) \quad (44)$$

The derivative of V with respect to time is as follow:

$$\begin{aligned}
&\dot{V} = u^T\dot{u} \\
&\dot{V} = u^T(I + M^T)(P_\Omega(u - (Mu + b)) - u) \\
&\dot{V} + u^T(I + M^T)u = u^T(I + M^T)P_\Omega(u - (Mu + b))
\end{aligned} \quad (45)$$

Hence, $\psi(u) = u^T(I + M^T)u$, which indicates the passivity of (20). ■, c.q.f.d.

G Analysis of Joint Sampling Interval Resolution

In Figure 21, we plot the performance of optimal SVM models trained on datasets with lower sampling resolutions $\{20\,\text{deg}, 22\,\text{deg}, 25\,\text{deg}, 30\,\text{deg}, 45\,\text{deg}\}$. This led to datasets of the following sizes $\{5.4m, 2.3m, 870k, 240k, 8.9k\}$, respectively. For each dataset, we performed the same CV procedure to find the optimal parameters C and σ, that yield the best error rates. The error rates reported in Figure 21, are from testing each model on the original 20 deg-sampled dataset. Clearly, as the joint sampling interval increases, the dataset size decreases dramatically, this leads to a same trend in performance. The most critical evidence of the fact that, randomly sampling a small percentage from the 20 deg-sampled dataset is not equivalent to increasing the joint sampling interval, is the constant decrease in FPR. For every sampling interval > 20 deg, the *optimal* SVMs cannot or *marginally* reach the error-rates achieved by the sub-sampled datasets (Figure 9). The rationale behind this evidence can be easily explained geometrically. With higher joint sampling intervals, less "collided" regions are explored in the robot configuration search procedure, hence, the negative class is not spanned properly and FPR decreases dramatically. On the other hand, by sampling joint configurations with a lower interval, we manage to represent each class properly, and a small randomly sampled percentage of the dataset is capable of exhibiting the underlying class distribution.

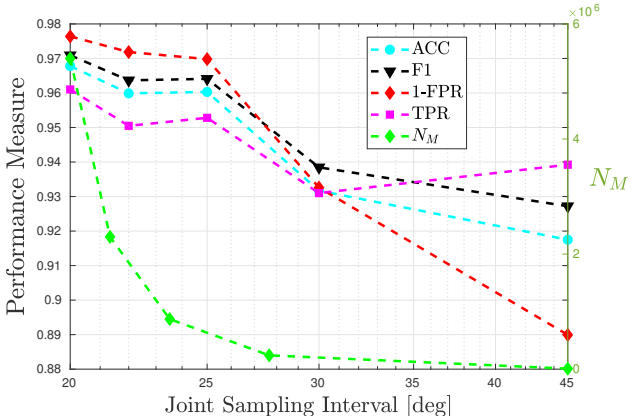


Figure 21. Joint sampling granularity tests. We learn optimal models (concerning classification error) on different datasets created with increasing joint sampling intervals: $\{20\,\text{deg}, 22\,\text{deg}, 25\,\text{deg}, 30\,\text{deg}, 45\,\text{deg}\}$. As the sampling interval increases, the dataset size decreases dramatically (from millions to thousands in two steps) as well as the FPR. N_M represents dataset size. The reported error rates are from the testing set of the 20 deg-sampled dataset.

H Selection of CPSP method

The two state-of-the-art sparse SVM approaches are: (i) that of (Cotter and Srebro 2013) , which we will name the Sub-Gradient (SG) method and (ii) the Cutting Plane Subspace Pursuit (CPSP) method introduced by (Joachims and Yu 2009), respectively. The SG approach is a very simple, yet theoretically motivated method to learn a sparse approximation $\tilde{\mathbf{w}}$ of $\mathbf{w}$, ensuring an empirical $0/1$ error

bound. From an initially dense estimation of $\mathbf{w}$, (Cotter and Srebro 2013) use a randomized classification rule to search for a $\tilde{\mathbf{w}}$ by minimizing a loss function $f(\tilde{\mathbf{w}})$ through sub-gradient descent. This approximation is bounded by $4||\mathbf{w}||^2$, hence it will select the optimal set of support vectors that comply with this bound. This method achieves the best approximation in terms of accuracy compared to other methods to date (Cotter and Srebro 2013). However, it will select as many support vectors as needed to achieve this accuracy, yielding a sparse but not bounded solution. Moreover, as mentioned earlier, the size of our dataset limits are capability of learning a dense solution from all the datapoints. Fortunately, the Cutting Plane Subspace Pursuit (CPSP) algorithm (Joachims and Yu 2009), specifically tackles these problems. Instead of finding the best sparse approximation ($\tilde{\mathbf{w}}$) from a dense one ($\mathbf{w}$), it directly estimates a solution to $\mathbf{w}$ with a strict bound on the number of support vectors, by reformulating the SVM optimization problem.

I Notation table, framework schematic and source codes

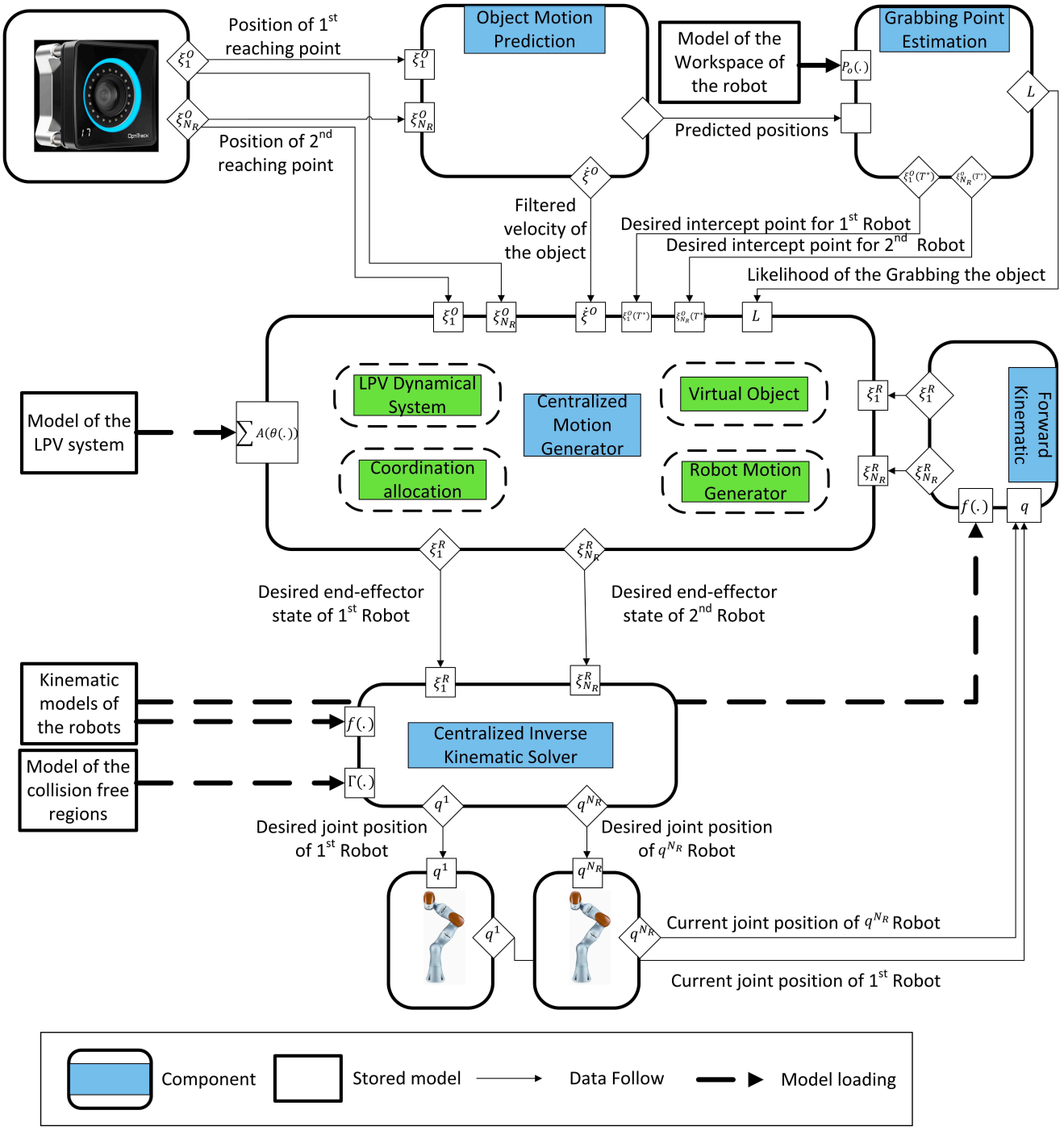


Figure 22. Block diagram for coordinated multi-arm motion planning for reaching a large moving object. Where N_R represents the total number of robot arms. T represents the motion prediction duration. In this paper, we assume that the low-level controller of the robot is a perfect tracking controller.

Table 2. Nomenclature

Variable	Domain	Definition
ε	$\in \mathbb{R}_{>0}$	A small positive number.
k	$\in \mathbb{R}_{>0}$	A large constant positive number.
$\mathbf{k}$	$\in \mathbb{R}_{>0}$	A constant positive number.
t	$\in \mathbb{R}_{>0}$	Time.
δ_j	$\in \mathbb{R}_{>0}$	Minimum likelihood threshold of j^{th} robot's workspace .
δ	$\in \mathbb{R}_{>0}$	Minimum joint likelihood threshold.
T^*	$\in \mathbb{R}_{>0}$	Time when the object is kinematically reachable.
N_R	$\in \mathbb{N}$	Number of the available robot arms.
N_{sv}	$\in \mathbb{N}$	Number of the support vectors.
N_M	$\in \mathbb{N}$	Number of the samples points.
τ_{c_i}	$\in \mathbb{R}_{(0,1)}$	Synchronization allocation parameter of i^{th} robot.
γ	$\in \mathbb{R}_{(0,1)}$	Coordination parameter.
d_n	$\in \mathbb{N}$	Dimension of the states of the virtual/real object or one robot.
d_N	$\in \mathbb{N}$	Dimension of the states of all the robots in total.
d_{s_i}	$\in \mathbb{N}$	Number of scheduling parameters of i^{th} robot.
d_{q_i}	$\in \mathbb{N}$	Number of the joints of i^{th} robot.
d_Q	$\in \mathbb{N}$	Number of the joints of all the robots.
d_{c_i}	$\in \mathbb{N}$	Number of the scheduling parameters of i^{th} robot.
q^i	$\in \mathbb{R}^{d_{q_i}}$	Joint angles of i^{th} robot.
q^{ij}	$\in \mathbb{R}^{d_{q_i}+d_{q_j}}$	Joint angles of the i^{th} and j^{th} robots.
q^i_j	$\in \mathbb{R}$	Angle of j^{th} joint of i^{th} robot.
$f(q^i_j)$	$\in \mathbb{R}^3$	Cartesian position of of j^{th} joint of i^{th} robot with respect to the world frame.
J_i	$\in \mathbb{R}^{d_n \times d_{q_i}}$	Jacobian matrix of i^{th} robot.
ξ^R_j	$\in \mathbb{R}^{d_n}$	Position of the j^{th} end-effector.
ξ^V_j	$\in \mathbb{R}^{d_n}$	Position of j^{th} reaching point on the *virtual object*.
ξ^V	$\in \mathbb{R}^{d_n}$	Position of the *virtual object*
ξ^O	$\in \mathbb{R}^{d_n}$	Position of the real object.
ξ^O_j	$\in \mathbb{R}^{d_n}$	j^{th} feasible reaching point on the real object.
ξ^d_j	$\in \mathbb{R}^{d_n}$	Static target of the asynchronous behavior of j^{th} robot.
${}^j\xi$	$\in \mathbb{R}^{d_n}$	ξ in the reference frame of j^{th} robot base.
θ^W_j		Set of GMM parameters of the workspace model of j^{th} robot.
Θ^W		Set of GMM parameters of the workspace model of all the robots.
x^V	$\in \mathbb{R}^{2d_n}$	States of the virtual object's dynamical system.
x^R_j	$\in \mathbb{R}^{2d_n}$	States of j^{th} end-effector .
x^d_j	$\in \mathbb{R}^{2d_n}$	States of the static target of the asynchronous behavior of j^{th} robot.
θ_j	$\in \mathbb{R}^{d_{s_j}}$	Scheduling parameters for pos./orient. dynamics for j^{th} robot.
A_{jk}, $\mathbf{A}_j(\theta_j)$	$\in \mathbb{R}^{2d_n \times 2d_n}$	Affine dependent state-space matrices for j^{th} robot.
P^R_i, Q_iR P^V, Q^V	$\in \mathbb{R}^{2d_n \times 2d_n}_{\succ 0}$	Auxiliary matrices which are used in the stability and convergence proofs.
$\Gamma(.)$	$\in \mathbb{R}$	Self-collision boundary.
$\dot{\theta}^-_j$, $\dot{\theta}^+_j$	$\in \mathbb{R}^{d_{q_j}}$	Conservative lower and upper bounds of the joint limits of j^{th} robot.
μ_p	$\in \mathbb{R}_{>0}$	Intensify coefficient in the IK solver.
η	$\in \mathbb{R}^{d_N}$	Dual decision vector.
ν	$\in \mathbb{R}$	Dual decision variable.
$k(.)$	$\in \mathbb{R}_{>0}$	RBF kernel.
σ	$\in \mathbb{R}_{>0}$	Kernel width.
y_i	$\in \{-1, 1\}$	Positive/negative labels.
α_i	$\in \mathbb{R}_{[0,C]}$	Weights of the support vectors.
C	$\in \mathbb{R}$	Penalty factor used in learning the SCA boundary.
c^j_i	$\in \mathbb{R}^3$	Center of the sphere on the i^{th} joint of the j^{th} robot.
r^j_i	$\in \mathbb{R}_{>0}$	Radius of the sphere on the i^{th} joint of the j^{th} robot.
b_-, b_+	$\in \mathbb{R}_{>0}$	Minimum/Maximum distance of the safe boundary.

Table 3. The implementation toolboxes which are provided by the authors.

Name	Purpose	Language	Uniform Resource Locator (URL)
Multiarm_ds	Centralized motion generator	C++	`https://github.com/sinamr66/Multiarm_ds`
QP_IK_solver	Centralized IK solver	C++	`https://github.com/sinamr66/QP_IK_solver`
SESODS_lib	Estimating parameters of second order LPV based DSs	Matlab	`https://github.com/sinamr66/SESODS_lib`
ML_toolbox	Machine learning toolbox	Matlab	`https://github.com/epfl-lasa/ML_toolbox`
SCA-Boundary-Learning	Learning SCA boundary	Matlab/C++	`https://github.com/nbfigueroa/SCA-Boundary-Learning`
SVMGrad	RBF kernel SVM and its Gradient	C++	`https://github.com/nbfigueroa/SVMGrad`
kuka-rviz-simulation	Simulation of the KUKA	C++	`https://github.com/epfl-lasa/kuka-rviz-simulation.git`
SCA_data_construction	Constructing data set for SCA	C++	`https://github.com/sinamr66/SCA_data_construction`